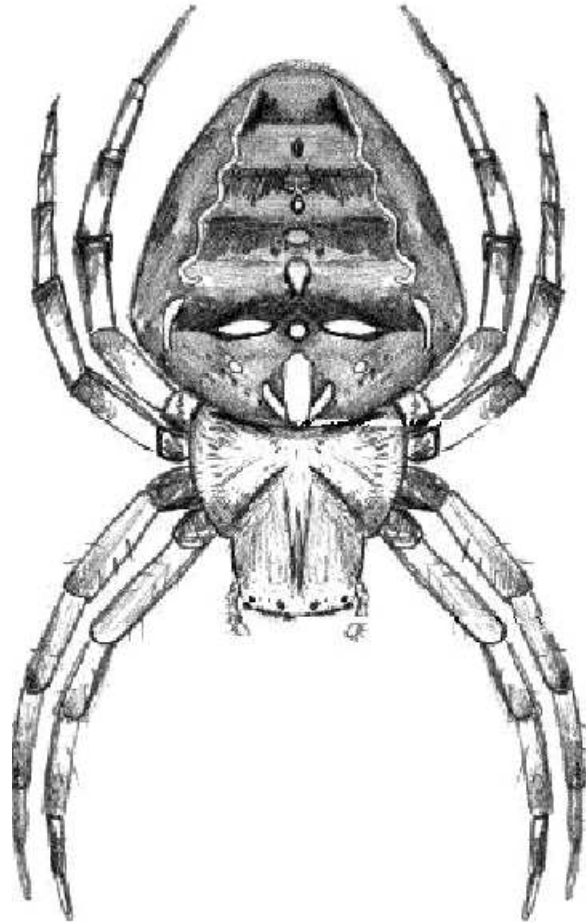

KumoNoSu

Version 1.0; May 17, 2006

USER'S MANUAL



Eric Hansen
Daniel Rypl
Victor Saouma
Department of Civil Engineering,
University of Colorado, Boulder
Boulder, CO 80309-0428

Under Contract from:
Tokyo Electric Power Service Company
3-3-3 Higashiueno, Taito-ku, Tokyo 110-0015

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THIS REPORT WAS PREPARED BY THE ORGANIZATION(S) NAMED BELOW AS AN ACCOUNT OF WORK SPONSORED OR COSPONSORED BY THE TOKYO ELECTRIC POWER SERVICE COMPANY (TEPSCO), TEPSCO ANY COSPONSOR, THE ORGANIZATION(S) NAMED BELOW, NOR ANY PERSON ACTING ON BEHALF OF ANY OF THEM:

(A) MAKES ANY WARRANTY OR REPRESENTATION WHATSOEVER, EXPRESS OR IMPLIED, (I) WITH RESPECT TO THE USE OF ANY INFORMATION, APPARATUS, METHOD, PROCESS OR SIMILAR ITEM DISCLOSED IN THIS REPORT, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR (II) THAT SUCH USE DOES NOT INFRINGE ON OR INTERFERE WITH PRIVATELY OWNED RIGHTS, INCLUDING ANY PARTY'S INTELLECTUAL PROPERTY, OR (III) THAT THIS REPORT IS SUITABLE TO ANY PARTICULAR USER'S CIRCUMSTANCES; OR

(B) ASSUMES RESPONSIBILITY FOR ANY DAMAGES OR OTHER LIABILITIES WHATSOEVER (INCLUDING ANY CONSEQUENTIAL DAMAGES, EVEN IF TEPSCO OR THEIR REPRESENTATIVES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES) RESULTING FROM YOUR SELECTION OR USE OF THIS REPORT OR ANY INFORMATION, APPARATUS, METHOD, PROCESS OR SIMILAR ITEM DISCLOSED IN THIS REPORT.

ORGANIZATION(S) THAT PREPARED THIS REPORT:

UNIVERSITY OF COLORADO AT BOULDER

ABSTRACT

KumoNoSu is a graphical front end to T3D (a finite element mesh generator), and T3D2Merlin (an structural analysis finite element analysis data preparation program).

KumoNoSu has been written explicitly for the Merlin finite element code, and as such has numerous built-in facilities to handle both cracks and/or dams. Nevertheless, the program is general enough to accommodate most other finite element codes.

KumoNoSu enables the user to interactively define the boundary of the structure to be meshed. Following each new boundary entity definition, the graphical display is updated. Once the boundary has been completely defined, then T3D is internally invoked, and a finite element mesh is generated. In the second module of KumoNoSu the user specifies the specific attributes of the finite element mesh (such as material properties, boundary conditions, incremental loads) with reference to the entity defining the boundary. This in turn will internally generate a complete input data file for the Merlin finite element code.

ACKNOWLEDGMENTS

KumoNoSu is a graphical front end to the mesh generator T3D and T3D2Merlin converter developed by Dr. Daniel Rypl.

KumoNoSu was originally developed by Dr. Eric Hansen, and subsequently modified/maintained by Dr. Gary Haussmann as a front end to the computer code MERLIN, through a contract from the Tokyo Electric Power Service Company (TEPSCO) with the Department of Civil Engineering of the University of Colorado in Boulder (Victor Saouma, P.I.).

The authors would like to acknowledge the numerous feedbacks, bugs reports and other constructive comments of: Guido Camata, Sonia Fortuna, C. Chang, Wiwat Puatsananon, and in particular of Takashi Shimpo and Yoshinori Yagome. It was through their numerous comments that KumoNoSu has matured into a solid, reliable, powerful and original program.

Contents

1	Introduction	12
1.1	Concepts of Boundary Representation	12
1.1.1	Hierarchy	13
1.1.2	Mesh Size/Density	14
1.2	Kumo Layout	14
2	File	16
3	Define Boundary	18
3.1	Vertex	19
3.2	Curve	20
3.2.1	Higher Order Curves	23
3.3	Patch	24
3.4	Surface	27
3.5	Region	28
3.6	Entity Groups	31
3.7	Mouse Vertex Creation	31
3.8	Mouse Curve Creation	31
3.9	Master/Slave	32
3.10	Embedded Reinforcement	32
3.11	Cracks	34
3.11.1	Crack Segments	34
3.11.2	Discrete Cracks	36
3.11.3	Examples	36
3.12	Crack Bridging a Truss Element	38
3.13	Crack Library	38
3.14	Elastic Boundary	39
3.15	Viscous Boundary	39
3.15.1	Discrete Dashpots/Nodal	39
3.15.2	Continuous Dashpots/Elements	39
3.16	Lumped Masses	40
3.16.1	Westergaard-Zangaar	40
3.16.2	User-Defined	41
3.17	Extrude	42
4	View	43
4.1	View Settings	43
4.1.1	Viewer Config	43
4.1.2	Selective Display	43
4.1.3	Domain Display	44
4.1.4	Load Display	44
4.1.5	Creation Control	44
4.2	Kumonosu Settings	45

4.3	Settings	45
4.4	Lighting	45
4.5	Reset Camera	45
5	Generate Mesh	47
6	T3D2MERLIN	48
6.1	Title	49
6.2	Keywords	49
6.2.1	General options	49
6.2.1.1	Monitor Max/Min Stress	49
6.2.1.2	AutoCrack	49
6.2.1.3	MultLDCurves	50
6.2.1.4	TimeAccelCurve	51
6.2.1.5	TimeDispCurve	51
6.2.1.6	TimeStressCurve	52
6.2.1.7	TimeStrainCurve	53
6.2.1.8	UserCurves	53
6.2.1.9	BandwidthMin	53
6.2.1.10	RealTimeView	54
6.2.1.11	Restart on Increment	54
6.2.1.12	Do Not Write Mesh	54
6.2.1.13	InitAnalflag	54
6.2.1.14	Split Output	54
6.2.1.15	Initial Temperature	54
6.2.2	Analysis Type Options	54
6.2.2.1	Implicit Transient	54
6.2.2.2	Explicit Transient	55
6.2.3	Strain Smoothing Options	55
6.2.4	Fracture Mechanics Options	57
6.2.5	Print Options	57
6.3	Material and Element Groups	58
6.4	AAR	58
6.5	Discrete/Continuum Groups	58
6.6	Eigenmode	58
6.7	Loads	60
6.7.1	Incremental Load Definition	60
6.7.1.1	Displacement BC's	60
6.7.1.2	Body Forces	62
6.7.1.3	Point Loads	62
6.7.1.4	Tractions	63
6.7.1.5	Centrifugal	64
6.7.1.6	Nodal Temperature	66
6.7.1.7	Point (xyz) Temperatures	66
6.7.2	Total Load Definition	66
6.7.2.1	Hydrostatic	66
6.7.2.2	Mud/Silt	66
6.7.2.3	Westergaard	67
6.7.2.4	Uplift	69
6.7.2.5	Dynamic Uplift	71
6.7.3	Reset Nodal Displacements	71
6.7.4	Heat Transfer/Seepage Loads	71
6.7.4.1	Temperature	71
6.7.4.2	Head	73

6.7.5	Dynamic Analysis	74
6.7.5.1	Harmonic Excitation	74
6.7.6	Convergence Control	75
6.7.6.1	Convergence Control	75
6.7.6.2	Solution Method	75
6.7.6.3	Convergence Acceleration	75
6.7.7	Solution Control	75
6.7.8	.pst File Control	76
6.7.9	Staged Construction/Excavation	76
6.8	Incremental Material Update	77
6.9	Generate Free Field	77
6.10	Run Free Field	80
6.11	Write .ctrl file	80
6.12	Generate Merlin .inp file	80
7	Merlin Files	81
A	MESH GENERATION	85
A.1	Introduction	85
A.2	Triangulation	85
A.2.1	Voronoi Polygon	86
A.2.2	Delaunay Triangulation	86
A.2.3	MATLAB Code	86
A.3	Finite Element Mesh Generation	87
A.3.1	Boundary Definition	87
A.3.2	Interior Node Generation	87
A.3.3	Final Triangularization	89
B	Rational Bezier Curve	90
B.1	Quadratic curve	91
B.2	Cubic curve	92
C	Examples of Finite Element Boundary Definition Output Files from PARSIFAL Program	95
C.1	Matrix and Inclusion without Interface elements	95
C.2	Matrix and Inclusion with Interface Elements	97
C.3	Matrix, Inclusion, and propagating Crack with Interface Elements	98
C.4	Matrix, Inclusion, and Two Propagating Cracks with Interface Elements	100
C.5	Matrix and Interior Crack with Interface Elements	102

List of Figures

1.1	Mesh Generation Process	12
1.2	KumoNoSu 's File Types	13
1.3	Entities Recognized by KumoNoSu	13
1.4	Hierarchy of Model Representation	14
1.5	Concept of Mesh Size	14
1.6	Concept of Mesh Count	14
1.7	Concept of Mesh Size; Curve and Patch	15
1.8	Concept of Factor	15
1.9	KumonoSu Toolbar	15
2.1	File Management	16
2.2	File Retrieval	17
3.1	Boundary Definition Menu	18
3.2	Vertex Definition	19
3.3	Vertex Rename Warning	20
3.4	Vertex Reorder/Duplicate Warning	21
3.5	Curve Definition	21
3.6	Curve Delete Warning	22
3.7	Curve Rename Warning	22
3.8	Circular Curve Definition	23
3.9	Elliptical Curve Definition	24
3.10	Parabolic Curve Definition	25
3.11	Hyperbola Curve Definition	25
3.12	Counter-Clockwise Patch Definition	26
3.13	Patch Definition	26
3.14	Patch Delete Warning	27
3.15	Surface Definition	27
3.16	Surface Definition	28
3.17	Surface Crack Definition	29
3.18	Surface Definition	29
3.19	Region Delete Warning	30
3.20	Region Definition	31
3.21	Vertex Definition by the Mouse	31
3.22	Curve Definition by the Mouse	32
3.23	Master-Slave Definition	32
3.24	Master-Slave Definition Example	33
3.25	Embedded Reinforcement	33
3.26	Crack Definition	34
3.27	Crack Orientation Definition for 2D Cases	34
3.28	Crack Orientation Definition for 3D cases	35
3.29	Discrete Crack Definition	35

3.30	2D Example of a Structure Crack	36
3.31	3D Example of a Structure Crack	37
3.32	2D Example of an Interface Crack, Upper and Lower	37
3.33	2D Example of an Interface Crack, Upper and Lower	37
3.34	3D Example of an Interface Crack	38
3.35	Rebar Crossing a Crack	38
3.36	Example of Rebar Crossing a Crack	39
3.37	Elastic Boundary	39
3.38	Viscous Boundary	40
3.39	Westergaard Added Mass Definition	41
3.40	Lumped Mass Definition	42
3.41	Extrude Interface	42
4.1	Viewer Configuration	43
4.2	Selective Display	43
4.3	Domain Display	44
4.4	Load Display	44
4.5	Creation Control	45
4.6	KumoNoSu Setting	45
4.7	Kumonosu Settings	46
4.8	Light Settings	46
5.1	Mesh Generation	47
6.1	T3D2Merlin Menu	48
6.2	Title Data Entry	49
6.3	Keywords Specification	49
6.4	Monitor Maximum Stress	50
6.5	Auto-Crack Specification	50
6.6	Load Displacement Curve Definition	51
6.7	Time Acceleration Curve Definition	51
6.8	Time Displacement Curve Definition	52
6.9	Time Stress Curve Definition	52
6.10	Time Strain Curve Definition	53
6.11	User Defined Curve	53
6.12	Data Entry for Transient Analysis	55
6.13	Computed Rayleigh Damping Coefficients	56
6.14	Data Entry for Explicit Transient Analysis	56
6.15	Material Input Data	58
6.16	AAR User Interface	59
6.17	Discrete ContinuumUser Interface	59
6.18	Eigenmode Specification	60
6.19	Loads Type	61
6.20		61
6.21	Definition of Body Forces	62
6.22	Point Load Definition	63
6.23	Traction Load Definition	63
6.24	Direction of Traction load	64
6.25	Centrifugal Load Definition	65
6.26	Nodal Temperature Load Definition	65
6.27	Point Temperature Load Definition	66
6.28	Hydrostatic Load Definition	67
6.29	Mud Silt Definition	68
6.30	Westergaards Added Mass Load Definition	68

6.31 Westergaards Orthogonal Added Mass Load Definition	69
6.32 FERC Uplift Loads	70
6.33 Uplift Load Definition	71
6.34 Reset (zero) Displacements Definition	72
6.35 Temperature Load Definition	72
6.36 Head Load Definition	73
6.37 Acceleration Specification	74
6.38 Indirect (COD) Solution Control	76
6.39 Suppress pst Output	76
6.40 Algorithm for Staged Construction	77
6.41 Algorithm for Staged Excavation	78
6.42 Graphical User Interface for Staged Construction/Excavation	78
6.43 Staged CONstruction/Excavation Warning if Insufficient number of Increments Specifiedn	78
6.44 Incremental material Update	79
6.45 Generate Free Field	79
6.46 Generate Merlin Input File	80
7.1 Merlin Toolbox	81
7.2 Program Interactions	82
7.3 Program Interactions	83
7.4 Program Interactions	83
A.1 Voronoi and Delaunay Tessellation	86
A.2 Control Point for a 2D Mesh	87
A.3 Control Point for a 3D Mesh	88
A.4 A Two Dimensional Triangularization AlgorithmControl Point for a 3D Mesh	88
B.1 Example of Rational Bezier Curve and Its Control Polygon	90
B.2 Elliptical Arc-Quadratic Curve	91
B.3 General Case of Elliptical Arc-Quadratic Curve	92
B.4 Elliptical Arc-Cubic Curve	93
B.5 Specific Case of Elliptical Arc-Cubic Curve	94
C.1 Matrix and inclusion without interface elements	95
C.2 Matrix and inclusion with interface elements	97
C.3 Matrix, inclusion, and propagating crack with interface elements	99
C.4 Matrix, inclusion, and two propagating cracks with interface elements	101
C.5 Matrix, Crack inclusion with interface elements	103

List of Tables

7.1 File Types	81
7.2 Hierarchy of Model Representatation	84

Chapter 1

Introduction

The finite element analysis requires the discretization of a structure into a mathematical representation using 1,2 or 3 dimensional elements.

The discretized structure is then subjected to the governing differential equation with essential (displacement) and natural (traction) boundary conditions.

A valid and well conditioned discretization of the structure being so important to the accuracy of the FEA solution that a mesh generator which produces a consistent, reproducible, high-quality mesh without great user intervention is key to a successful FEA.

Fig. 1.1 is a simple illustration of a physical problem, its boundary description, and the resulting finite element mesh.

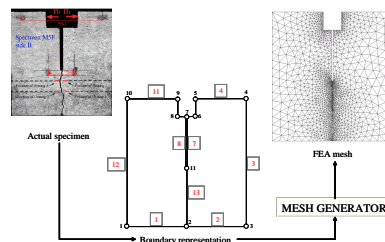


Figure 1.1: Mesh Generation Process

KumoNoSu is a graphical front end to two programs (both written by Dr. Daniel Rypł):

T3D a powerful mesh generator which can produce both unstructured (based on Delaunay triangulation) and structured meshes.

T3D2Merlin which enables definition of material properties, boundary conditions and loads for a Merlin input file.

Hence, KumoNoSu first produces a boundary definition of the physical object to be discretized, a `tt .bd` file. Once the boundaries of the solid are delineated and mesh-size generation guidelines established for the code, the program constructs a mesh to describe the structure, this is the `.t3d` file. Next, material properties and loading data must be added for each element to the geometric data set. This is saved into a `.ctrl` file. Finally, a finite element analysis algorithm may process the expanded data set into the Merlin `.inp` file, Fig. 1.2.

1.1 Concepts of Boundary Representation

The role of the boundary representation for T3D is as a geometrical description of individual model entities and representation of their topological relationships, Fig. 1.3:

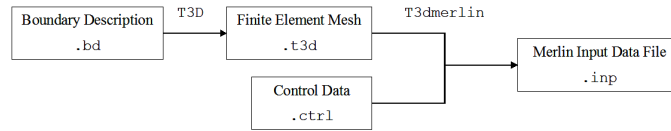


Figure 1.2: KumoNoSu 's File Types

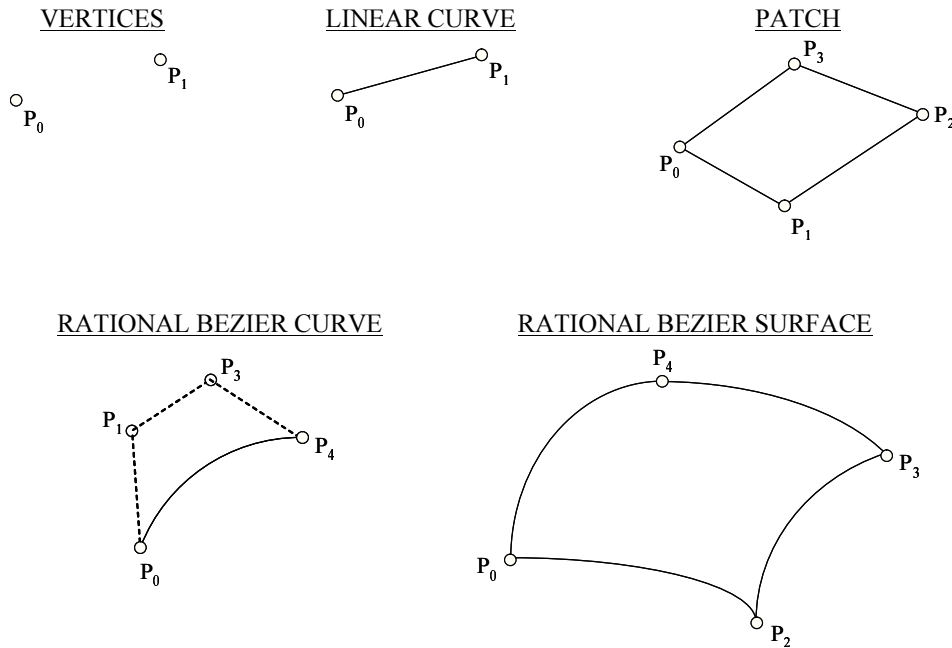


Figure 1.3: Entities Recognized by KumoNoSu

Vertices points in (x,y,z) space.

Curves defined by 2 end vertices, may be linear, quadratic, or cubic.

Patches planar collection of curves.

Surfaces non-planar, defined by (4) curves.

Shells non-planar collection of curves.

Regions set of non-self-intersecting boundary surfaces, patches, and shells.

1.1.1 Hierarchy

Those entities are defined hierarchically, Fig. 1.4 Lower level entities which belong to higher level entities are called the **Minor** entities of that higher level entity. Hence:

- Vertices are the minor entities of curves.
- Vertices and curves are the minor entities of patches and surfaces.
- Vertices, curves, surfaces, and patches are the minor entities of regions.

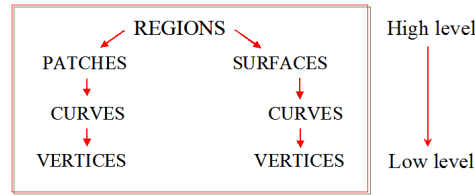


Figure 1.4: Hierarchy of Model Representation

1.1.2 Mesh Size/Density

In the process of generating a finite element mesh, it is highly desirable that one can control the mesh gradation or density. Hence, wherever the strain energy gradient is highest (such as in zones of stress concentration), we have a refined mesh. For unstructured meshes, this can be controlled by the **size** of an entity for triangular (2D) or tetrahedral (3D) meshes. In Fig. 1.5 we note how the reduction of the **size** parameter results in denser uniform mesh over the entire patch.

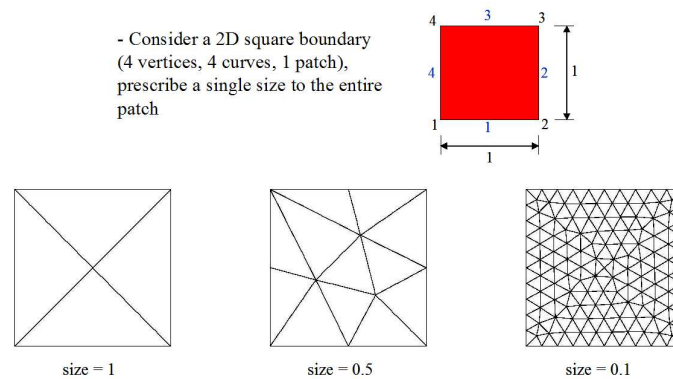


Figure 1.5: Concept of Mesh Size

For structured meshes, composed of quadrilaterals in 2D or hexagonal in 3D, mesh density can be controlled by the **count** concept, Fig. 1.6.

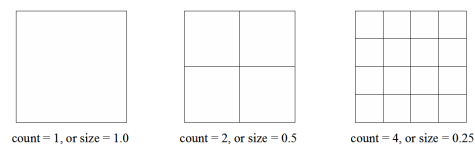


Figure 1.6: Concept of Mesh Count

Finally, **gradation** of the mesh can be accomplished by assigning different sizes to different entities (small size in the areas of denser mesh, larger size in areas of coarser mesh), Fig. 1.7.

1.2 Kumo Layout

KumoNoSu toolbars has the following main components, Fig. 1.9:

File Controls input/output with external files, Fig. 2.1. This is defined in described in 2

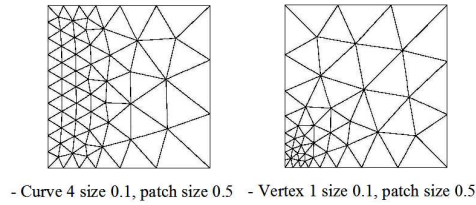


Figure 1.7: Concept of Mesh Size; Curve and Patch

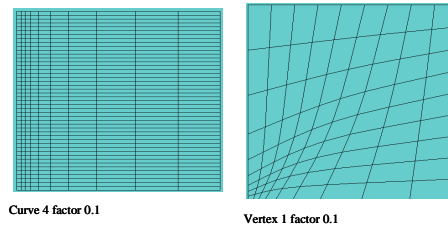


Figure 1.8: Concept of Factor

Define Boundary Allows user to define the boundary representation of the structure to be meshed, Chapter 3.

View Controls a number of viewing parameters, Chapter 4.2.

Generate Mesh to instruct T3D to generate the finite element mesh using the previously defined .bd file, Chapter ??.

T3D2Merlin As the main interface which allows definition of the material properties, loads, boundary conditions, Chapter 6.

Help For setup and info about the code.

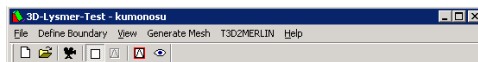


Figure 1.9: KumonoSu Toolbar

each one of them will be described in a separate chapter.

Chapter 2

File

The user clicks the file menu to initiate file management, Fig. 2.1 If the session is the initial session for

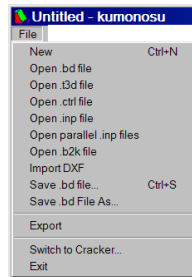


Figure 2.1: File Management

the project, there is no need to select the **File** task.

New Wipes out the memory, and enable the definition of a new bd file.

Open bd file If the user has previously created a boundary file and wishes to run another trial using the same boundary, the user should Open boundary file. KumoNoSu will present the user with the available .bd-files stored in the default directory. Even if stored in directories other than the default directory, other .bd-files are available by directory search, Fig. 2.2.

Open t3d file This will allow the user to open an existing mesh generated by t3d. The user may not modify the mesh defined in this file.

Open ctrl file This file contains the load and material properties information used by T3d2Merlin to generate a Merlin.inp file.

Open inp file to retrieve a Merlin file. There are few instances where this is needed.

Open Parallel input file Eanables the user to select one of the multiple files previously generated by KumoNoSu through its domain decomposition algortihm.

Open b2k file Opens a file coming from Beaver.

Import dxf file Import a dxf file (AutoCad). KumoNoSu will attempt to map vertices and curves. Surfaces and Regions (3D entities) are not recognized by KumoNoSu and will have to be manually defined.

Save bd file Once the boundary has been named, the user may save the entity at any phase of development.

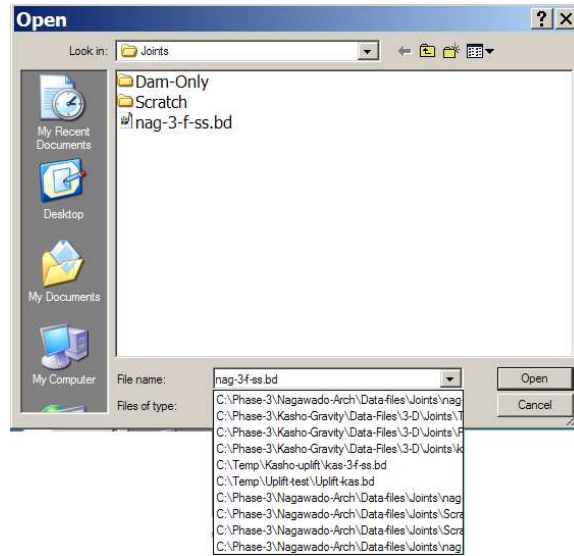


Figure 2.2: File Retrieval

Save bd file as Save a bd file for the first time, or with a new name.

Export Will create an .eps, .emf, .jpg, .bmp or .gif file out of the current main window display.

Switch to Cracker which enables the automatic simulation of crack propagation in 2D only

Chapter 3

Define Boundary

The Define Boundary module enables the user to define all the entities which describe the boundary of the structure to be discretized. Note this section deals only with the geometric definition of the structure, loads, material properties will be specified later.

This module, Fig. 3.1 will generate a .bd file which in turn will be executed by the T3D module (developed by Daniel Rypl).

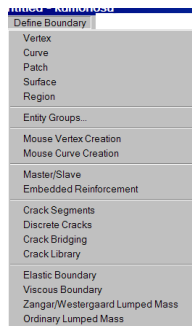


Figure 3.1: Boundary Definition Menu

Vertex to define individual points (called vertices), Sect. 3.1.

Curves are one dimensional curves (or lines) connecting vertices, Sect. 3.2.

Patch Define planar entities composed of 3 or more curves, Sect. 3.3.

Surfaces Define non planar entities composed of 3 or 4 curves, Sect. 3.4.

Regions which are three dimensional objects defined in terms of patches or region, Sect. 3.5.

Entity Groups To lump together various basic entities for easier reference later (such assign a traction to multiple patches), Sect. 3.6.

Mouse Vertex Creation Using a background grid, define vertices with the mouse, Sect. 3.7.

Mouse Curve Creation Using existing vertices, define new curves, Sect. 3.8.

Master/Slave: to tie force entities (vertices, curves, patches or surfaces) to have identical displacements, Sect. 3.9.

Embedded Reinforcement to define steel reinforcement perfectly bonded to the surrounded continuum, Sect. 3.10.

Crack Segments can be line or surface discontinuities, with or without interface elements, Sect. 3.11.1.

Discrete Cracks Define crack entities from previously defined crack segments, Sect. 3.11.2.

Crack Bridging If a reinforcement crosses a crack, Sect. 3.12.

Crack Library (not yet implemented)

Elastic Boundary To apply elastic springs at a vertex, along a curve, or on a patch/surface, Sect. 3.14.

Viscous Boundary To apply a nodal or continuum dashpot at a vertex, along a curve, or on a patch/surface, Sect. 3.15.

Zangaar/Westergaard Lumped Mass To determine and apply added masses along a curve or on a patch/surface, Sect. 3.16.1.

Ordinary Lumped Mass Manually defined, Sect. 3.16.2.

Extrude to 3D To extrude a two dimensional mesh into a three dimensional one, Fig. 3.17.

3.1 Vertex

To define a vertex, the user must specify a vertex number (note vertex numbers need not be sequentially numbered).

To edit an existing vertex, its id must be entered, and then user must click on **Edit**, Fig. 3.2.

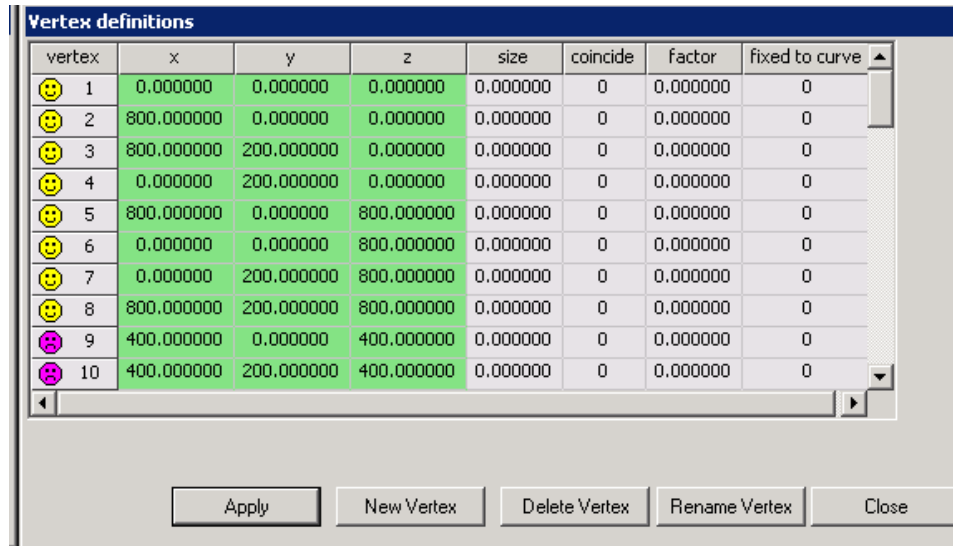


Figure 3.2: Vertex Definition

Mandatory information for vertices:

Vertex id Not necessarily sequential.

Coordinates Vertex coordinates are entered (for a two-dimensional model, the z-coordinate field is left blank) in the grid table.

Optional information:

Size The Size field allows the user to specify a dimension to elements in the vicinity of the vertex. The Size assigned to a vertex takes precedence over the Size assigned during patch definition (discussed later). This value can be left blank/zero.

Coincide Vertex Define the vertex number that is coincident with the current vertex. This option is most often associated with crack definition. Note that if two (or more) vertices have the same coordinates, the vertices with higher numbers are coincident to the vertex with the lowest number. A coincident vertex should have been previously defined.

Factor is a mesh size multiplying factor (default is one, even if the table shows zero) applied to the `-d` value defined as default for the T3D mesh generator. Hence, whereas **Size** specifies an absolute size for the mesh (irrespective of the `-d` value defined later), **Factor** is relative to that value.

Fixed to Curve Curve number that the vertex is fixed to. If a vertex is to be located on a curve, but not part of the connectivity of that curve, it is fixed to the curve. It should be noted that by default a vertex fixed to a parent entity inherits mesh size from that entity.

Control keys:

Apply will accept all changes.

New Vertex will generate a new blank row for data entry.

Delete Vertex will delete that entity. Careful, user must check if this vertex is not used in a curve.

Rename a vertex Allows the user to assign a new vertex id. Kumo will inform user if that vertex is used by curves, Fig. 3.3.

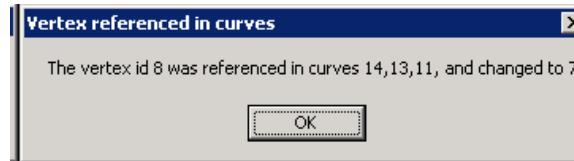


Figure 3.3: Vertex Rename Warning

Close will simply close the current GUI.

Note:

1. Table can be sorted in ascending/descending order of any of the column values, Fig. 3.4.
2. Duplicate vertex numbers are highlighted.
3. the “smiley” icon means that the vertex is used (at least once) in a curve definition. whereas the “sad” icon means that the vertex is an “orphan” and is not referenced by any curve. (This may happen if the vertex is Fixed on a curve or patch, and is used to have its displacements monitored).
4. User can use the usual Ctrl-C and Ctrl-V to copy and paste into the matrix.

3.2 Curve

Curve is a compulsory keyword. Mandatory information for all curves, Fig. 3.5: Fig. 3.5.

Curve id : Required for patch, surface, and shell definitions; does not have to be sequential.

Vertex definitions							
vertex	x	y	z	size	coincide	factor	fixe
10	400.000000	200.000000	400.000000	0.000000	0	0.000000	
9	400.000000	0.000000	400.000000	0.000000	0	0.000000	
7	800.000000	200.000000	800.000000	0.000000	0	0.000000	
7	0.000000	200.000000	800.000000	0.000000	0	0.000000	
6	0.000000	0.000000	800.000000	0.000000	0	0.000000	
5	800.000000	0.000000	800.000000	0.000000	0	0.000000	
4	0.000000	200.000000	0.000000	0.000000	0	0.000000	
3	800.000000	200.000000	0.000000	0.000000	0	0.000000	
2	800.000000	0.000000	0.000000	0.000000	0	0.000000	
1	0.000000	0.000000	0.000000	0.000000	0	0.000000	

Apply New Vertex Delete Vertex Rename Vertex Close

Figure 3.4: Vertex Reorder/Duplicate Warning

Curve definitions									
Curve	from	to	Size	Factor	Coincide	Count	Duplicate	Order	ID element
1	1	2	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
2	2	3	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
3	3	4	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
4	4	1	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
5	5	2	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
6	5	6	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
7	6	1	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
14	7	3	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
12	7	6	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
13	7	7	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
11	7	5	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>
15	7	4	0.000000	0.000000	0	0	0	Linear	<input type="checkbox"/>

Apply New Curve Delete Curve Rename Curve Close

Figure 3.5: Curve Definition

from-to : Curve start and end vertex numbers (The direction of the curve is from the Vertex i to Vertex j). While the user may assign any direction to the curve, he would be well advised to utilize a convention that facilitates COUNTER-CLOCKWISE element (a patch in two-dimension models) construction.

Optional information

Size The Size field allows the user to specify a dimension to elements in the vicinity of the curve. The Size assigned to a curve takes precedence over the Size assigned during patch definition (discussed later). This value can be left blank/zero.

Factor is a mesh size multiplying factor (default is one, even if the table shows zero) applied to the `-d value` defined as default for the T3D mesh generator. Hence, whereas **Size** specifies an absolute size for the mesh (irrespective of the `-d value` defined later), **Factor** is relative to that value.

Coincide Defines a curve number that is coincident with the current curve. This option is most often associated with crack definition. Note that if two (or more) curves have the same coordinates, the curve with higher id is coincident to the curve with the lower id. A coincident curve should have been previously defined.

Count : Define number of quad/hexa elements along the curve (for structured mesh only).

Duplicate : Define that the mesh nodes along this curve must duplicate those of the specified curve.

Curve order : Linear (order 2, default), Quadratic (order 3), Cubic (order 4). Circular, parabolic, hyperbolic and elliptical curves can be defined with curves of order 3 and 4; though the fourth order should be used preferably.

1D element : Specify that this curve is to be included in the mesh as a 1D element. 1D elements (typically steel reinforcement) requires a material ID, specified in the Material ID box.

Control keys:

Apply will accept all changes.

New Curve will generate a new blank row for data entry.

Delete Curve will delete that entity. Prior to deleting, KumoNoSu will check if this curve is not used subsequently, [3.6](#)



Figure 3.6: Curve Delete Warning

Rename Curve Allows the user to assign a new vertex id. Kumo will inform user if that curve id is already used, Fig. [3.7](#).



Figure 3.7: Curve Rename Warning

Close will simply close the current GUI.

3.2.1 Higher Order Curves

T3D (and thus KumoNoSu) uses rational Bezier curves¹ which are described in details in chapter B

When higher order curves (such as quadratic or cubic) are to be specified a second dialog appears for further definition of control points for the nonlinear curve, KumoNoSu will automatically calculate the control points for (4) types of curves:

Circular arcs : User must specify the center coordinates of the arc, and indicate if the arc is smaller or greater than 180° . Note that vertices i and j would have been defined in the previous dialog box, Fig. 3.8.

If all three points lie on a straight line, then the arc sustains an angle of 180° , and then a direction vector $(V_x, V_y, [V_z])$ defines the direction from the circle origin through the center of the arc.

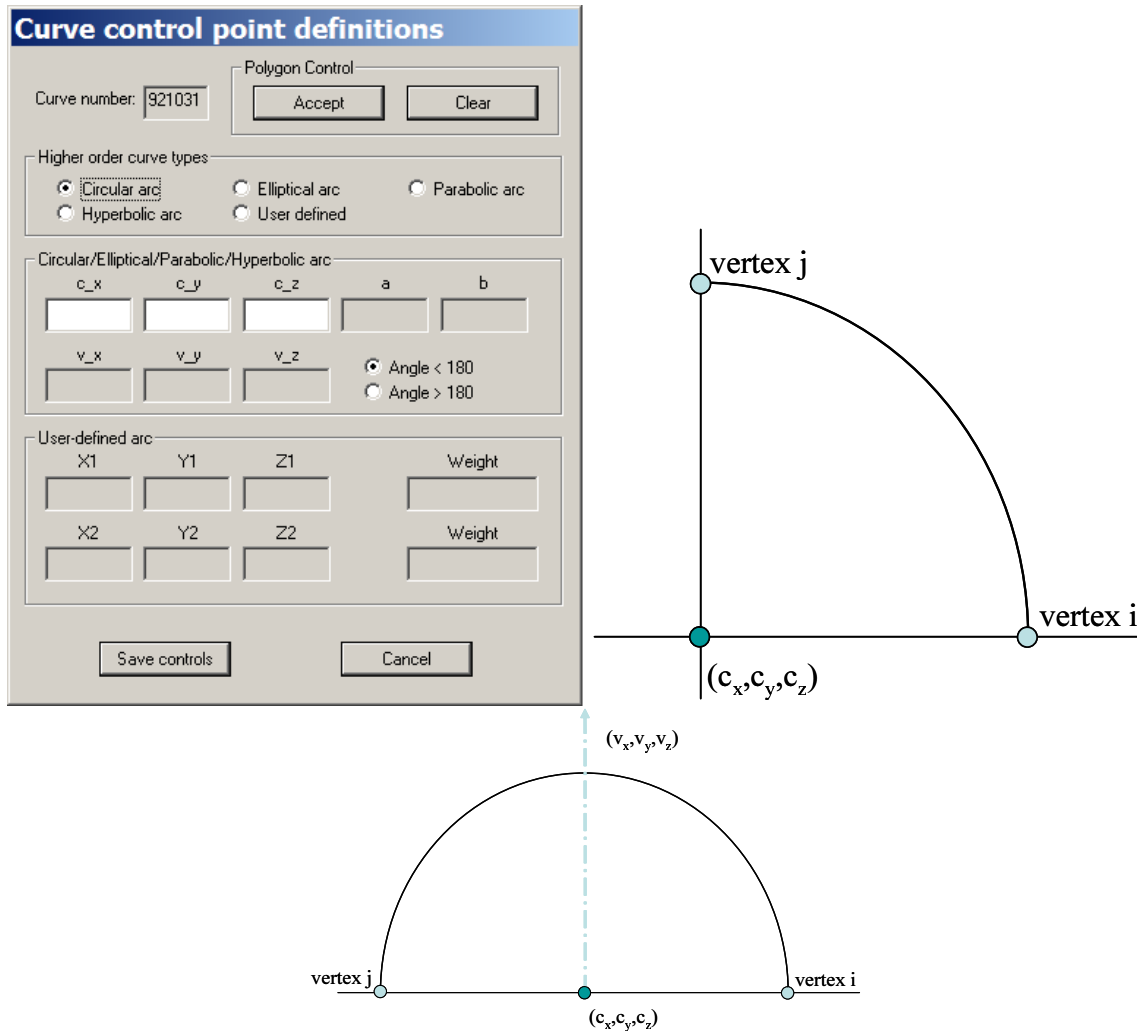


Figure 3.8: Circular Curve Definition

Elliptical arcs : Defined by the coordinates of the ellipse center $(c_x, c_y, [c_z])$, a and b and the direction vector $(V_x, V_y, [V_z])$ from the center to the center of the arc. Note that vertices i and j would have been defined in the previous dialog box, Fig. 3.9.

¹Rypl, D., *T3D, Triangularization of 3D Domains*, User Guide, January 2001.

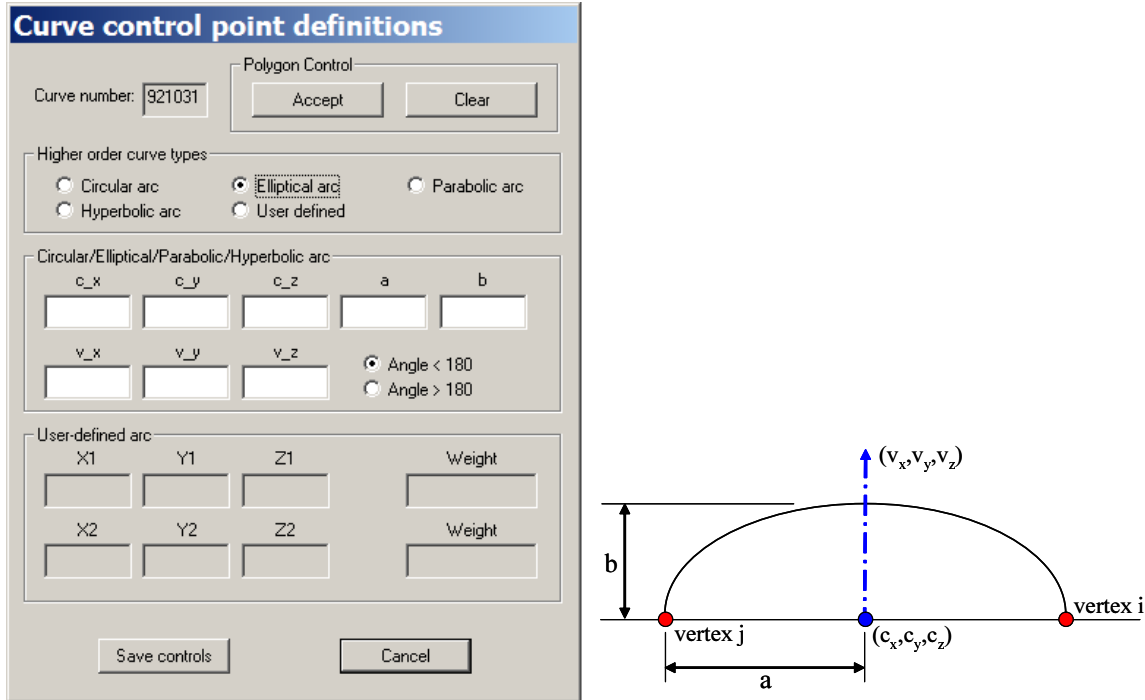


Figure 3.9: Elliptical Curve Definition

Parabolic arcs : The parabola is completely defined by the two vertices i and j (defined in the previous dialog box), and the parabola vertex $c_x, c_y, [c_z]$, Fig. 3.10.

Hyperbolic arcs : The hyperbola requires definition of a and b as well as a direction vector $(V_x, V_y, [V_z])$. Note that vertices i and j would have been defined in the previous dialog box, Fig. 3.11.

Additionally, control points and weights may be defined by the user.

For subsequent editing of the curve parameters, user should double click on the curve id. Note that if a curve is not linear, the smiley/sad face would have a different color.

3.3 Patch

A patch is a two dimensional section in a plane. A patch can be defined by multiple curves, however all those curves must be co-planar (if not an unforeseen error may occur).

During Patch definition, the direction of the defined members is critical. This direction must be **counterclockwise** with respect to the normal vector (automatically determined by KumoNoSu), and pointing outward.

If the curve definition (from i to j vertex) follows the counter-clockwise orientation, it is considered positive in the boundary curve definition. If the curve definition is oriented clockwise, it is considered negative in the boundary curve definition, Fig. 3.12. Once the data has been accurately added or adjusted for a patch, the user selects the Accept option from the Patch Control sub-menu. Once all vertex data has been entered accurately, the user selects the Save patches button at the bottom of the Patch menu-board.

Mandatory information for all patches, Fig. 3.13: Fig. 3.5:

Patch id number Required for reference in 2D; required for region definitions in 3D; does not have to be sequential.

Boundary curves List of curves completely defining the patch.

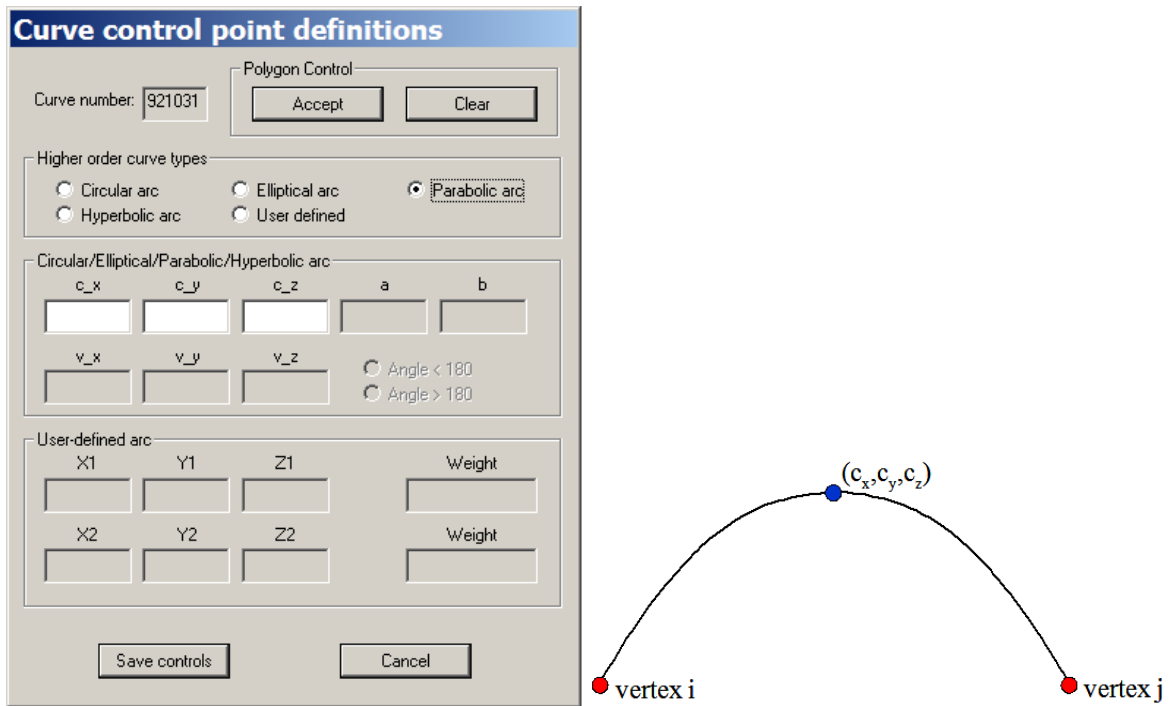


Figure 3.10: Parabolic Curve Definition

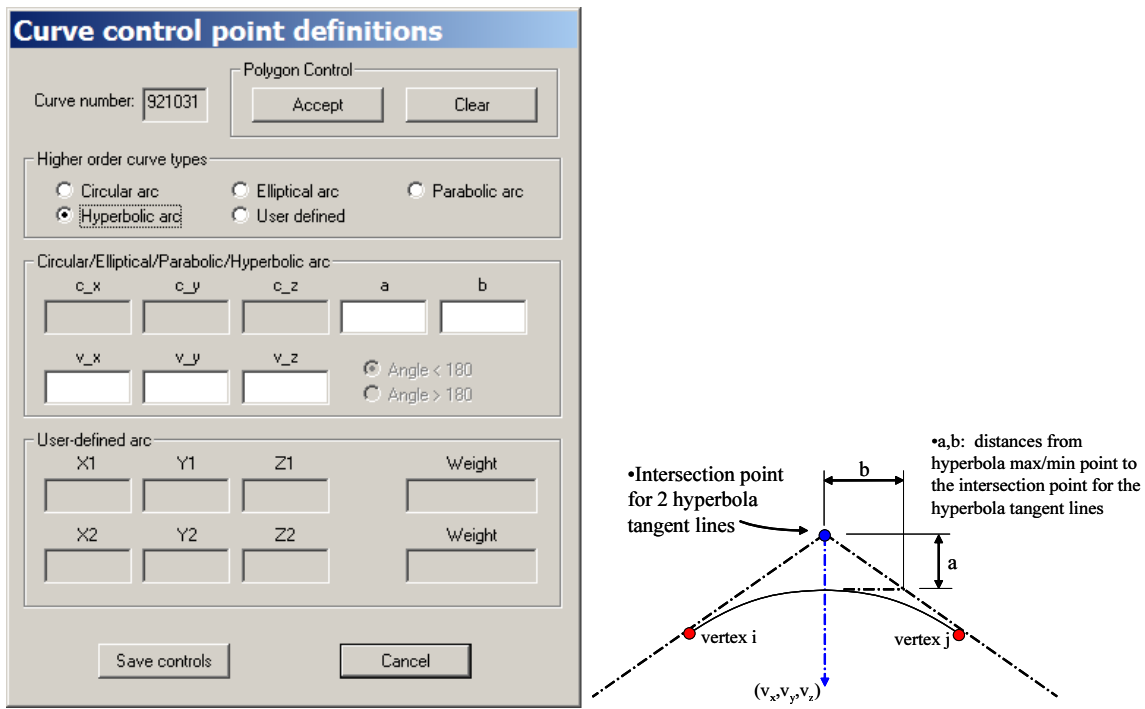


Figure 3.11: Hyperbola Curve Definition

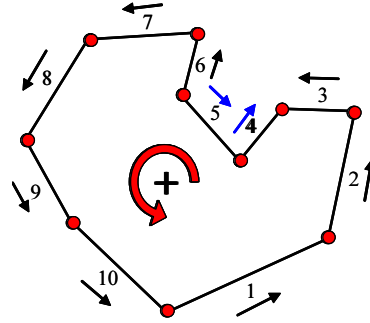


Figure 3.12: Counter-Clockwise Patch Definition

Define Patches														
Patch	Curves										Material	Size	Coincide	Hole
	1	2	3	4	5	6	7	8	9	10				
😊 1	13	12	-6	-11							0	0.000000	0	<input type="checkbox"/>
😊 2	-7	-12	15	4							0	0.000000	0	<input type="checkbox"/>
😊 3	-1	-4	-3	-2							0	0.000000	0	<input type="checkbox"/>
😊 4	5	2	-14	11							0	0.000000	0	<input type="checkbox"/>
😊 5	14	3	-15	-13							0	0.000000	0	<input type="checkbox"/>
😘 6	7	1	-5	6							0	0.000000	0	<input type="checkbox"/>

Apply New Patch Delete Patch Rename Patch Close Max Curves: 10

Figure 3.13: Patch Definition

Material Material number associated with the current patch (only for 2D models). Material number must be assigned to two-dimensional patches. The material properties associated with the Material number are defined later from the T3D2Merlin pull-down menu using the Element Groups menu.

Optional selections:

Size Defines mesh size for a patch. The Size field for a patch defines the strong dimension of elements within the vicinity of the patch. The Size assigned within the Patch definition menu is weaker than the previous two Size definitions and always defers to previous definition.

Coincide Defines patch number that is coincident with the current patch. For crack modeling in three-dimensional space, Coincide Patch defines the two surfaces initially occupying the same space.

Hole Defines that this patch is a hole (it will not contain a mesh). This is used for inserting holes in larger patches.

Additional optional parameters may be entered by double clicking on the patch id:

Factor Defines patch mesh size multiplication factor.

Fixed Vertices Defines vertex id of those vertices which lie inside the patch (if any exist).

Fixed Curves Defines curve id of those curves which lie inside the patch (if any exist).

Subpatches Defines the patch numbers of those patches which lie completely inside the patch. For example, hole patches are subpatches (they are inside larger patches).

Boundary curves No. 2 Defines a second set of boundary curves (used for special circumstances only).

Control keys:

Apply will accept all changes.

New patch will generate a new blank row for data entry.

Delete patch will delete that entity. Prior to deleting, KumoNoSu will check if this curve is not used subsequently, [3.14](#)



Figure 3.14: Patch Delete Warning

Rename Curve Allows the user to assign a new patch id. Kumo will inform user if that curve id is already used.

Close will simply close the current GUI.

Max Curves Allows the user to increase the number of allowable curves which define a patch. Careful, if you decrease the current number, you may lose data associated with patch having a large number of curves.

3.4 Surface

A surface is bound by three or four curves (not necessarily coplanar) defined sequentially (i.e. each curve must be connected to the previously defined one, and the one defined after it).

Curves are specified as positive integers, i.e. we need not worry about continuity of the curves for the surface definition. Surface is positive if curves are ordered clockwise around the outer side of the surface.

Curve order in the surface definition matters. Curve direction does not matter.

When defining a region, a surface id may be +ve (if its outward direction is pointing out), or ve (if its outward direction is pointing inside).

Surfaces are only applicable to 3D boundary descriptions, Fig. [3.15](#).

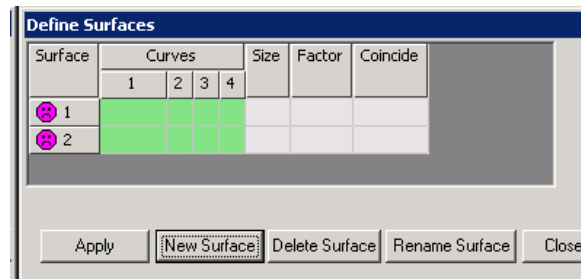


Figure 3.15: Surface Definition

Input data for the Surfaces, are of two types:

Mandatory for all surfaces:

Surface id Required for region definitions in 3D; does not have to be sequential.

Surface curves Three or four curves which define the four sides of the surface. At least one of these curves must be nonlinear, else the surface is planar and is a patch.

Optional

Size Defines mesh size for the surface.

Factor Defines surface mesh size multiplication factor.

Coincide Defines surface number that is coincident with the current surface.

In certain instances, polygon control information will be required for the surface, in which case a second dialog box will open for this information.

It should be noted that the order of the four curves which compose the surface is dependent upon the outer side of the surface (i.e. the side that defines the exterior of the model. Hence, curves are ordered clockwise around the outer side of the surface, and the following rules apply, Fig. 3.16.

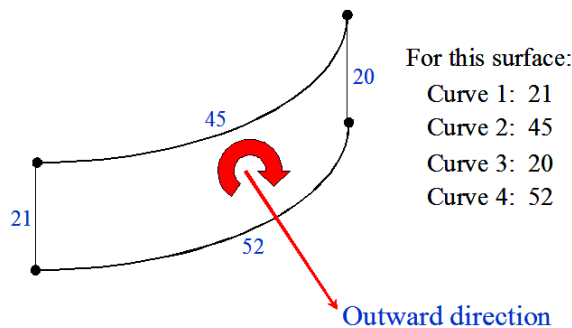


Figure 3.16: Surface Definition

1. If only 1 or 2 **nonconsecutive** curves in a surface are order 3, do not specify a polygon for the surface.
2. If 2 or more **consecutive** curves in a surface are order 3, a polygon must be specified for the surface.
3. If only 1 or 2 **nonconsecutive** curves in a surface are order 4, do not specify a polygon for the surface.
4. If 2 or more **consecutive** curves in a surface are order 4, four polygon coordinates must be specified for the surface.
5. If 1 curve is order 3, and the **previous** or **subsequent** curve is order 4, two polygon coordinates must be specified.

When defining cracks, bounded by two adjacent surfaces then: The two surfaces must be identically defined, i.e the first vertex of the first curve must have the same coordinates for both surfaces, Fig. 3.17.

3.5 Region

In three-dimensional models the regions must be defined. A region defines a volume. Just as curves comprise boundaries define patches, patches and/or surfaces combine to define the boundaries of a region.

For a region, a patch is defined as positive if its normal points in a positive direction, as defined by the global coordinates. A negative sign during the Boundary patch definition must precede any patch that has a negative unit normal.

The Size defined in the Region definitions is weaker still than the Size defined in the Patch definitions. The user edits regions in a similar fashion as vertices and patches: insert the appropriate number and double click the field box.

Mandatory input parameters for all regions: 3.18.

Region id For reference; does not have to be sequential.

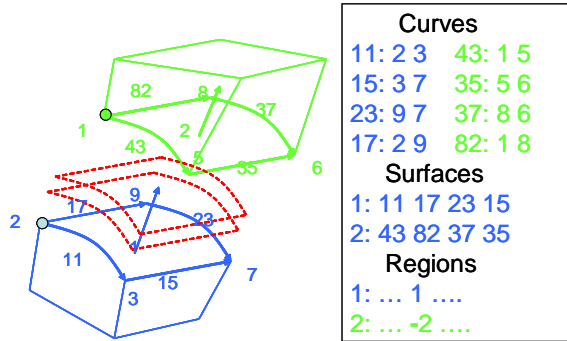


Figure 3.17: Surface Crack Definition

Define Regions						
Region	8704	8705	8603	8604	8605	8502
Material	1	1	1	1	1	1
Hexahedral?	Hexa	Hexa/tetra	Hexa	Hexa	Hexa	Hexa
Size	160.00000	160.00000	160.00000	160.00000	80.00000	160.00000
Factor	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Hole	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Patches						
1	-7248	-7258	-6238	-6248	-6258	-5228
2	7249	7259	6239	6249	6259	5229
3	62489	62589	52389	-624890	-625890	42289
4	-897	-898	-896	52489	52589	-895

Max Patches: 195
 Max Surfaces: 3

Apply New Region Delete Region Rename Close

Figure 3.18: Surface Definition

Material Material number associated with the current region.

Hexahedral User may specify if the elements to be generated in the current regions are Tetrahedrons (default), Hexahedral, or a combination of the two.

Boundary patches List of patches composing the boundary of the region (if any exist).

Boundary surfaces List of surfaces composing the boundary of the region (if any exist).

Optional entries are

Size Define mesh size for the region.

Factor Define region mesh size multiplication factor.

Fixed curves Define any curves which lie inside the region and are not connected to any patch or surface.

Hole Indicating that we have a three-dimensional cavity.

Control keys:

Apply will accept all changes.

New Region will generate a new blank row for data entry.

Delete Region will delete that entity. Prior to deleting, KumoNoSu will provide the user with a list of all entities pertaining to this region, 3.19. The user can then delete all those entities, or uncheck those which should be retained.

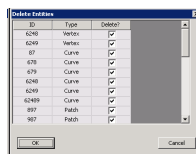


Figure 3.19: Region Delete Warning

Rename Allows the user to assign a new patch id. Kumo will inform user if that curve id is already used.

Close will simply close the current GUI.

Max Patches/Surfaces Allows the user to increase the number of allowable patches/surfaces which define a patch. Careful, if you decrease the current number, you may lose data associated with patch having a large number of curves.

Certain important rules apply for region definition, Fig. 3.20:

1. Boundary surface numbers are always positive.
2. Boundary patch numbers are positive if the patches normal points OUT of the region.
3. Boundary patch numbers are negative if the patches normal points INTO the region.
4. Since the surface normal (defined by the order of the surface curves) is always out of the region, boundary surface numbers are always positive.

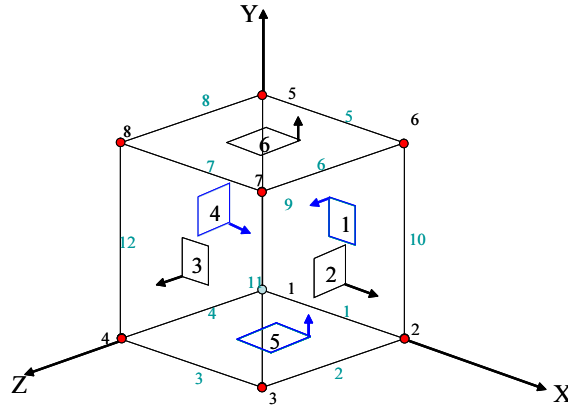


Figure 3.20: Region Definition

3.6 Entity Groups

NEED TO FIX KUMO Entity groups enable the user to lump or group together various entities which will subsequently inherit the same characteristic (particularly in the load definition).

3.7 Mouse Vertex Creation

User can use the mouse to define new vertices and have them connected by curves, Fig. 3.21. However, user must first select a projection plane, and then specify the third coordinate through the slider. At that point, when the left button of the mouse is pressed a new vertex is created at the closest grid point. Grid resolution must be defined within the View Option in Section 4.1.5. Note that the mouse current position is echoed in the bottom toolbar. If **Enable Point Creation** is active, then pressing the mouse creates the new vertex. If **Connect Points with Curves** is active, then sequential curves are defined.

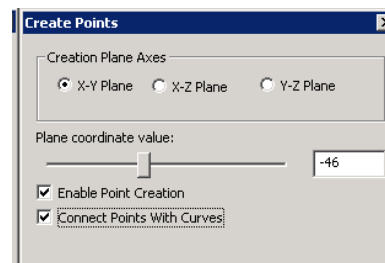


Figure 3.21: Vertex Definition by the Mouse

3.8 Mouse Curve Creation

This option, Fig. 3.22 enables the user to select individual vertices with the mouse and have them connected by curves.

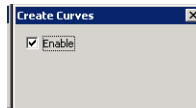


Figure 3.22: Curve Definition by the Mouse

3.9 Master/Slave

Master/slave nodes in the FE mesh may be defined in KumoNoSu through the master/slave pair definition dialog. Both vertex pairs and curve pairs may be defined in the dialog. In the case of M/S curve pairs, all nodes in the FE mesh that coincide with this curve will be M/S nodes, Fig. 3.23. Mandatory

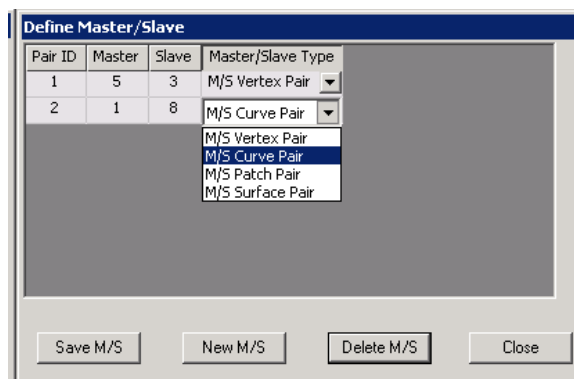


Figure 3.23: Master-Slave Definition

information:

Master/Slave counter This id will never be explicitly referenced, it is just a counter

Master/Slave pairs.

Entity type Specify if the M/S pair is a vertex, curve, patch, or surface pair (patch and surface pairs in 3D only).

NOTE: all vertices or curves defined as M/S must also be defined as **coinciding** entities.

Fig. 3.24 is an example of master slave definition.

3.10 Embedded Reinforcement

Embedded reinforcement, to be added to the MERLIN input file, may be defined in KumoNoSu through the reinforcement definitions dialog, Fig. 3.25.

Two methods are available for embedded reinforcement: definition by existing vertices or definition by end coordinates. Either of these options may be selected in the 'Definition type' section. If definition by vertices is chosen, the information is imputed in the 'Rebar vertices information'. The material number for the reinforcement, beginning vertex number ($Vert_i$), and ending vertex number ($Vert_j$) must be defined. If definition by coordinates (x, y, z) is selected, the information is inputted in the 'Rebar (x, y, z) information' section. The material number, (x, y, z) coordinates for the beginning (i) vertex, and (x, y, z) coordinates for the end (j) vertex must be defined. After the requisite information is imputed, click 'Accept' to save the reinforcement definition. When all rebars have been defined, click 'Save reinf' to save the information and close the dialog box.

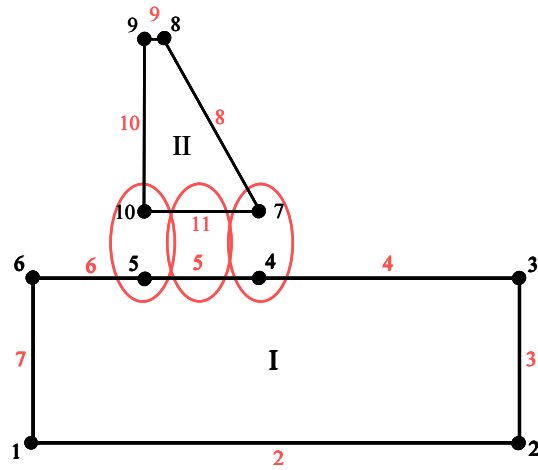


Figure 3.24: Master-Slave Definition Example

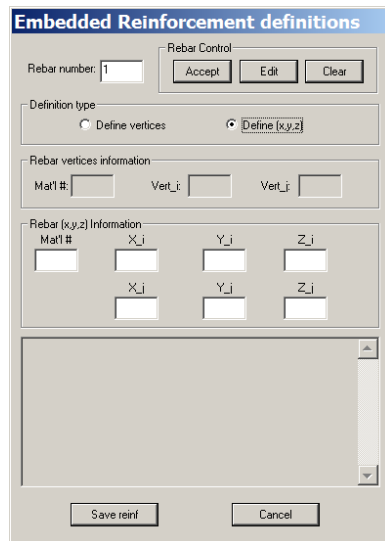


Figure 3.25: Embedded Reinforcement

3.11 Cracks

3.11.1 Crack Segments

A Crack is composed of one or multiple segments. The first step consists in defining those pairs of segments which will later constitute a crack, Fig. 3.26:

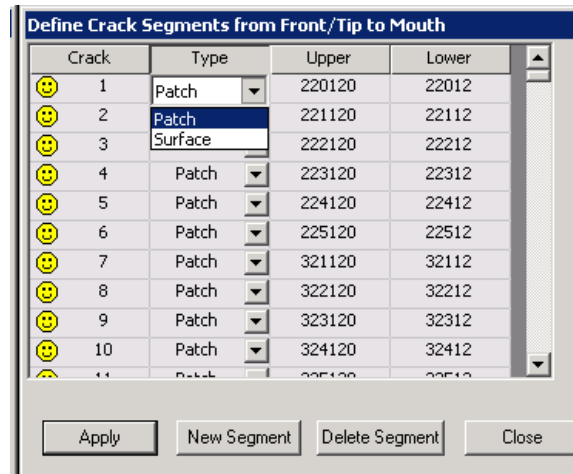


Figure 3.26: Crack Definition

Type which can be a curve, patch or surface.

Upper/Lower entities.

Note:

1. Crack paths are always defined from crack front to crack mouth.
2. Curve orientation (i and j vertices) in each segment must be from crack front to crack mouth.
3. The two curves or patches composing each crack segment must be defined as **Coincide**.
4. 2D Crack segment upper and lower curves are prescribed according to the location of the crack front and Fig. 3.27.

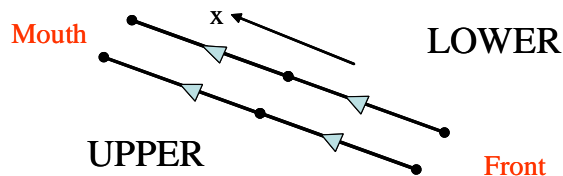


Figure 3.27: Crack Orientation Definition for 2D Cases

5. 3D Crack segment upper and lower patches are prescribed according to which patch is on top when the crack opens (in terms of the positive global coordinate directions), Fig. 3.28.

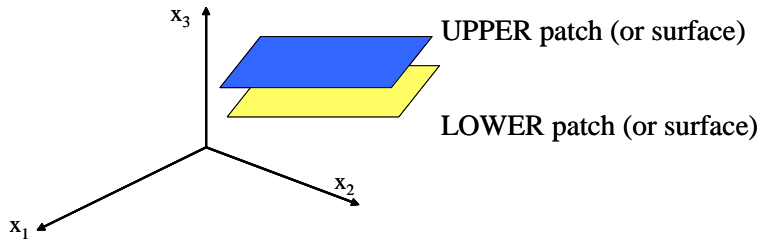


Figure 3.28: Crack Orientation Definition for 3D cases

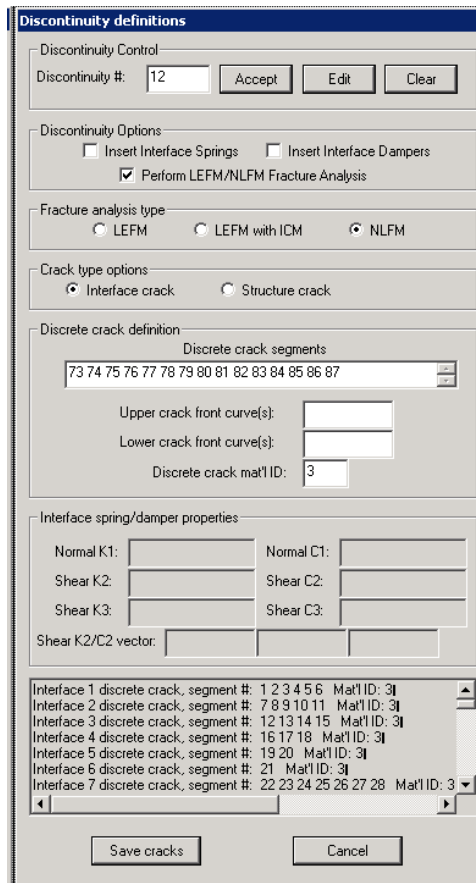


Figure 3.29: Discrete Crack Definition

3.11.2 Discrete Cracks

Once the crack segments have been defined, the User may now define the discrete cracks, Fig. 3.29.

Discontinuity id is the id associated with the crack/discontinuity about to be defined.

Discontinuity Option **Partially implemented**

Insert Interface Spring between the two lips of the discontinuity. **Not yet implemented.**

Insert Interface Damper between the two lips of the discontinuity. **Not yet implemented.**

Perform LEFM/NLFM Analysis, is by default the option.

Fracture Analysis Type Linear Elastic Fracture Mechanics (LEFM), Nonlinear Fracture Mechanics (NLFM), or LEFM with interface crack elements (cohesive crack model).

Crack Type Option Interface cracks are between two patches in two-dimensional analyses and between two regions in three-dimensional analyses. A structure crack develops within a patch or a region.

Discrete Crack Definition

Discrete crack segments lists the crack segments id's constituting the discrete crack.

Upper crack front curve Vertex (or curve in 3D) at the upper surface crack front.

Lower crack front curve Vertex (or curve in 3D) at the lower surface crack front.

Discrete Crack Mat'l ID to be used only if interface elements are to be inserted along the crack (NLFM or LEFM with ICM options).

Interface spring/damper properties Inactive.

Note that identification of the upper and lower curves is for the proper application of the uplift (FERC model).

3.11.3 Examples

Fig. 3.30 is an example of 2D structure crack where the Upper crack surface is curve 6, the lower one is 7, the crack front is 7. The crack is defined by only one crack segment.

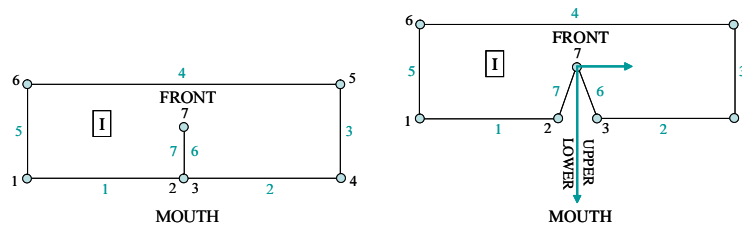


Figure 3.30: 2D Example of a Structure Crack

Fig. 3.31 is an example of 3D structure crack. The crack segment is composed of the pair of patches I and II, the crack front is defined by curve 5. The upper segment is II (because it is “above” patch I in the positive X direction), the lower one is I.

A 2D interface crack is shown in Fig. 3.32. Vertex 11, 12, 13 must coincide with vertex 1, 2, 3 respectively (Vertex 11, 12, 13 must be defined AFTER vertex 1, 2,3). Curves direction MUST be defined from tip (or front) to the mouth of the crack Curves 14 and 15 must coincide with curves 4 and 5 respectively. (Curves 14 and 15 must be defined AFTER curves 4 and 5).

Another example of 2D interface crack is shown in Fig. 3.33.

Fig. 3.34 is an example of 3D interface crack.

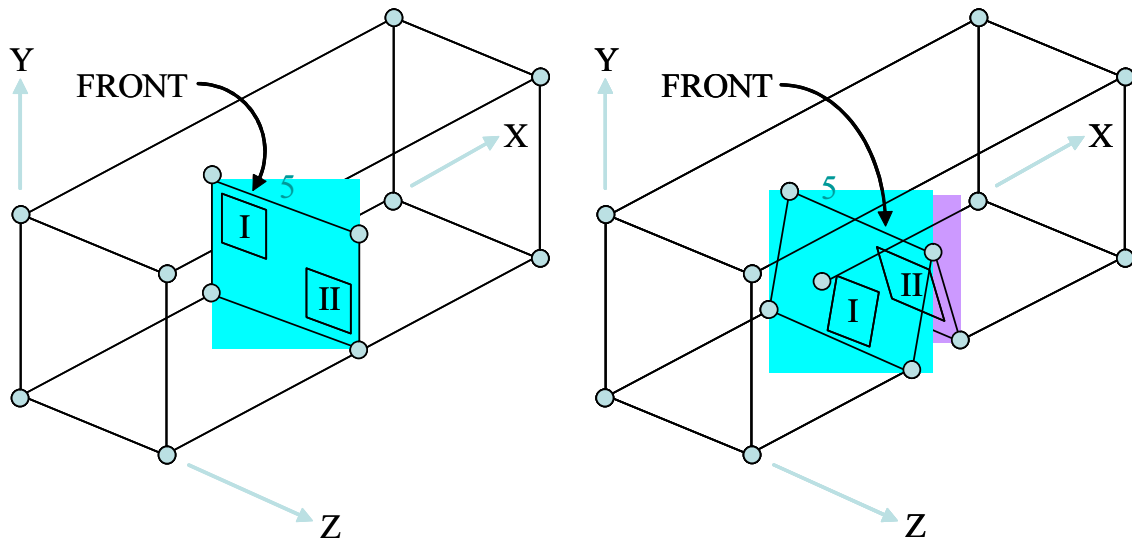


Figure 3.31: 3D Example of a Structure Crack

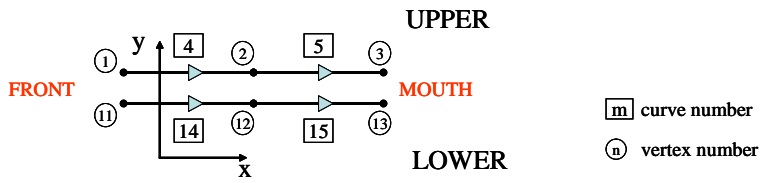


Figure 3.32: 2D Example of an Interface Crack, Upper and Lower

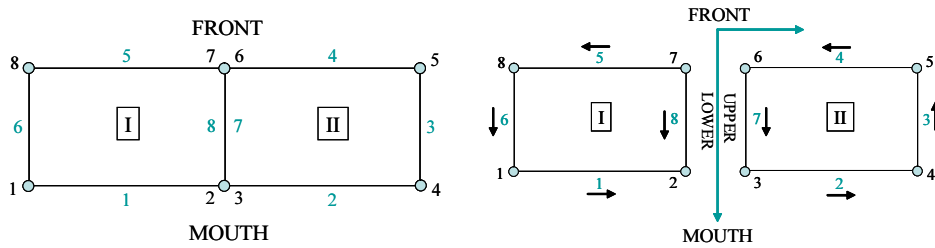


Figure 3.33: 2D Example of an Interface Crack, Upper and Lower

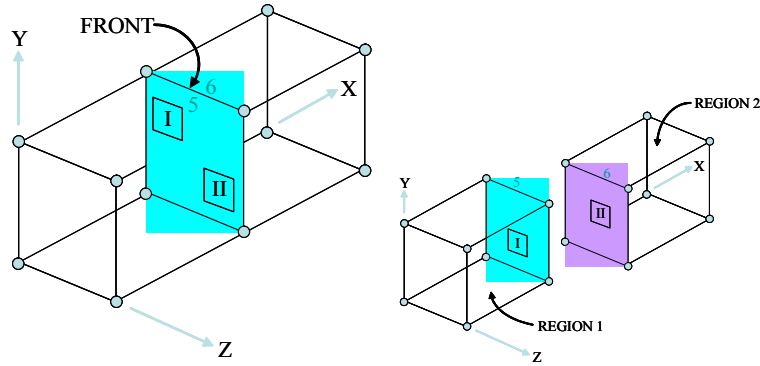


Figure 3.34: 3D Example of an Interface Crack

3.12 Crack Bridging a Truss Element

In reinforced concrete analysis, a crack may cross a truss element modeling a rebar, Fig. 3.35.

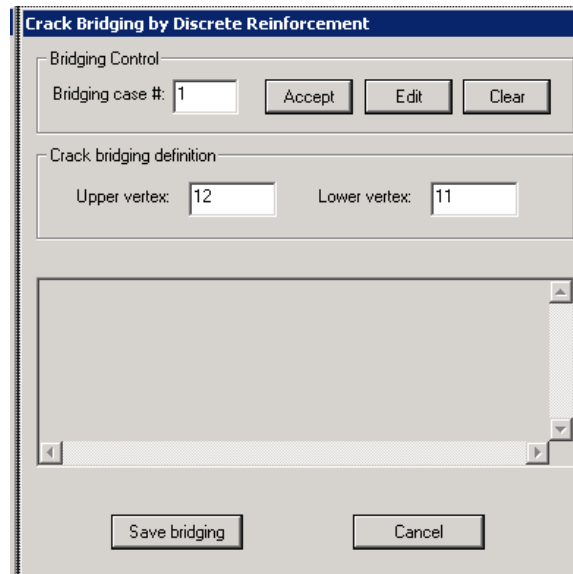


Figure 3.35: Rebar Crossing a Crack

Case No. For reference; does not have to be sequential.

Upper vertex Vertex number of the curve end which lies on the upper surface of the crack.

Lower vertex Vertex number of the curve end which lies on the lower surface of the crack

Fig. 3.36 is a simple illustrative example.

3.13 Crack Library

Inactive

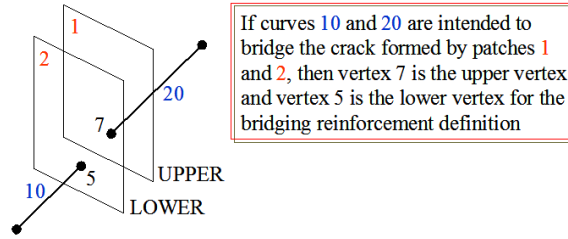


Figure 3.36: Example of Rebar Crossing a Crack

3.14 Elastic Boundary

Elastic Boundary enables the user to specify elastic springs along a global direction (X , Y or Z) on a vertex, along a curve or over a patch.

The spring stiffness can be determined either from

$$K = \frac{AE}{t} \tag{3.1}$$

where A is the tributary area of the spring (automatically determined by KumoNoSu) on the basis of adjacent elements, E is the Young’s modulus of the adjoining material (automatically detected by KumoNoSu), and t is the effective *thickness* to be specified by the user, or can be explicitly given, Fig. 3.37.

Spring ID	Type	Entity ID	Use Stiffness?	Depth	Stiffness	DOF
1	Patch	1	Depth	12		X
2	Surface		Stiffness		23	Y
3	Patch		Depth			X
4	Patch		Depth			X
5	Patch		Depth			X
6	Patch		Depth			X
7	Patch		Depth			X
8	Patch		Depth			X
9	Patch		Depth			X
10	Patch		Depth			X
11	Patch		Depth			X
12	Patch		Depth			X
13	Patch		Depth			X
14	Patch		Depth			X

Figure 3.37: Elastic Boundary

User is reminded that the spring connects an entity to a rigid support.

3.15 Viscous Boundary

3.15.1 Discrete Dashpots/Nodal

3.15.2 Continuous Dashpots/Elements

Viscous boundaries absorbs the energy from pressure and shear waves, (?). Damping of these waves is based upon the elastic properties of the continuum along the boundary

$$C = \rho V_i A \begin{cases} \text{Shear Wave: } V_s = \sqrt{\frac{G}{\rho}} \\ \text{Pressure Wave: } V_p = \frac{1}{s} V_s; s^2 = \frac{1-2\nu}{2(1-\nu)} \end{cases} \tag{3.2}$$

Mandatory information for all viscous boundaries, Fig.3.38

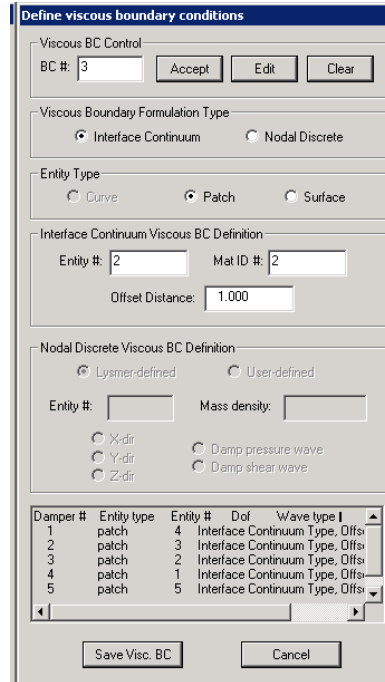


Figure 3.38: Viscous Boundary

B.C id boundary condition number, for reference.

Formulation Interface continuum will generate continuum dashpots elements along the selected entity. In 2D these will be four noded elements, and in 3D 6 or 8 noded elements.

Entity type (curve in 2D, patch or surface in 3D).

Interface Continuum parameters

Entity No. to identify the curve, patch or surface.

Mat ID Material group associated with the viscous interface elements (to be defined later).

Offset Distance The interface dashpot elements have a zero thickness formulation. Hence, user can select an arbitrary offset distance for visualization.

Nodal Discrete parameters

Entity No. number of the entity at which to apply viscous B.C.s.

Mass density mass density γ of the material on the viscous boundary (if zero or blank, the γ of the actual material will be used).

X-dir, Y-dir, [Z-dir] degree of freedom to be damped.

Damp pressure wave/Damp shear wave wave to be damped.

3.16 Lumped Masses

3.16.1 Westergaard-Zangaar

Mandatory information for all Westergaard (vertical upstream face) or Zangaar (inclined upstream face) added mass, Fig. 3.39

Figure 3.39: Westergaard Added Mass Definition

Nodal mass number for reference.

Lumped Mass type Westergaard or Zangar.

Entity type Curve (in 2D) or Patch (in 3D).

Entity No. entity number.

Axis defining reservoir depth X , Y (or Z in 3D).

K constant K constant for Westergaard equation (defined by Westergaard as 51.0 lb/ft^3 or 8011.4 N/m^3).

Water elevation Elevation of the reservoir surface (note that this is not the relative depth of the reservoir, but rather the elevation of the surface).

Water modulus Elastic modulus of the water (may be taken as 300 kips/in^2 or 2.068 GPa).

Fluid weight weight per unit volume of the fluid (Westergaard only).

Accel of gravity Acceleration of gravity.

Quake period Period of the earthquake motion (Westergaard only).

Relative fluid depth Relative depth of the reservoir.

3.16.2 User-Defined

This feature must be checked.

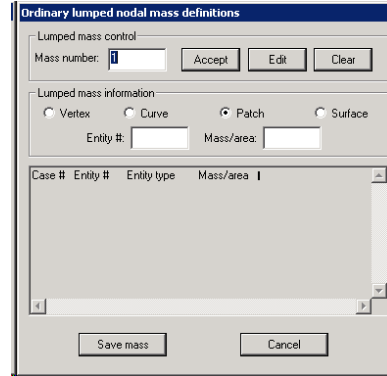


Figure 3.40: Lumped Mass Definition

3.17 Extrude

The extrude capability, Fig. 3.41 enables the user to begin with a 2D mesh definition, and then extrude it into a 3D mesh along the z axis by a user specified length.

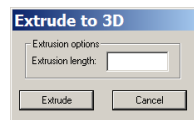


Figure 3.41: Extrude Interface

Chapter 4

View

4.1 View Settings

4.1.1 Viewer Config

Fig. 4.1...

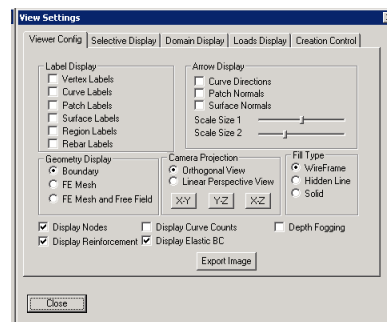


Figure 4.1: Viewer Configuration

4.1.2 Selective Display

Fig. 4.2

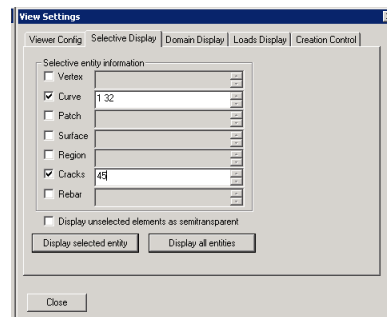


Figure 4.2: Selective Display

4.1.3 Domain Display

Fig. 4.3 ...

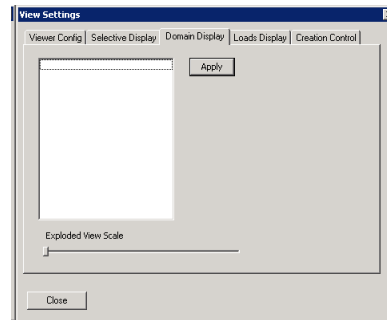


Figure 4.3: Domain Display

4.1.4 Load Display

Fig. 4.4

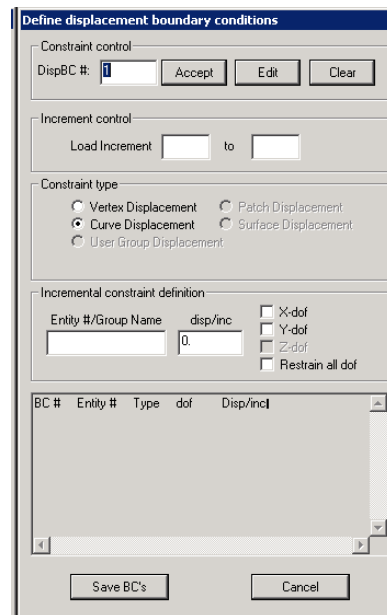


Figure 4.4: Load Display

4.1.5 Creation Control

Fig. 4.5

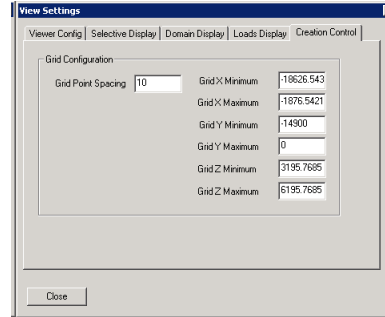


Figure 4.5: Creation Control

4.2 Kumonosu Settings

4.3 Settings

The user may wish to set or adjust the Preprocessor settings, Fig. 4.6. This enables the user to specify:

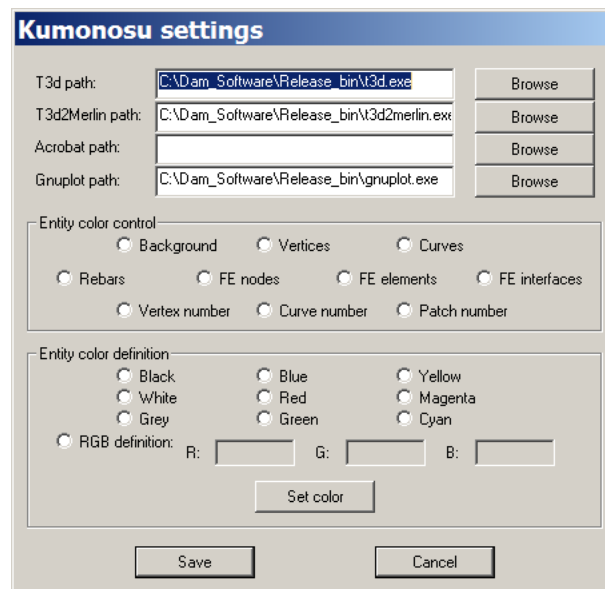


Figure 4.6: KumoNoSu Setting

1. Paths where the supporting software resides (T3d, T3d2Merlin, Acrobat Reader and Gnuplot).
2. Colors of various entities used by KumoNoSu .

4.4 Lighting

Fig. 4.8 ...

4.5 Reset Camera

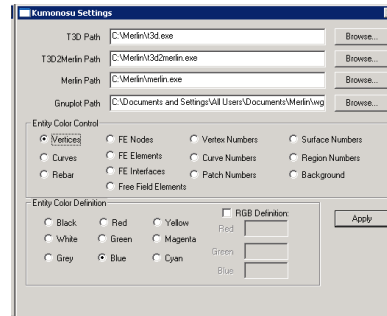


Figure 4.7: Kumonosu Settings

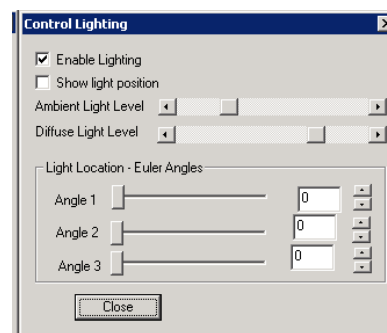


Figure 4.8: Light Settings

Chapter 5

Generate Mesh

The Generate Mesh dialog box, Fig. 5.1 offers the user the opportunity to modify the default mesh size and the coincide tolerance.

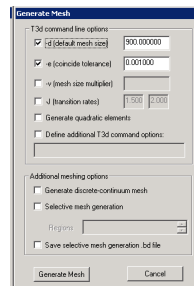


Figure 5.1: Mesh Generation

d indicates to the code a mesh size for nodes, curves, patches and regions that have not had their respective sizes explicitly set.

e represents the allowable tolerance for coincident nodes. Two nodes within this specified tolerance will be treated as coincident nodes.

v [check](#)

-j [check](#) for tetra/hexa?

Quadratic elements [buggy](#)

Additional T3D commands can be specified. Please consult the T3D manual.

Discrete-Continuum will place an interface element in between all the edges of the elements.

Selective Mesh Generation enables the user to test the mesh by meshing only selected regions.

Save selective mesh .bd file for testing/debugging purposes.

Chapter 6

T3D2MERLIN

After successful generation of the mesh by T3d (.t3d file), the user must first define a control file (.ctrl) and then run T3d2Merlin to create a Merlin input file. All of the information required by the control file may be defined from the T3d2Merlin pull-down menu. The menu, Fig. 6.1 lists:

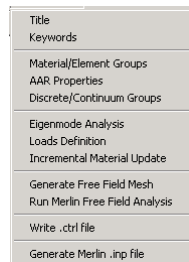


Figure 6.1: T3D2Merlin Menu

Title Sect. 6.1.

Keyword to specify the control commands, Sect. 6.2.

Material/Element Groups For material definition, Sect. 6.3.

AAR Properties Sect. 6.4.

Discrete/Continuum Groups To enable the generation mesh with interface elements in between all the elements, Sect. 6.5.

Eigenmode Analysis Sect. 6.6.

Loads Definition Sect. 6.7.

Incremental Material Update to modify material properties within load increments, Sect. 6.8.

Generate Free Field for dynamic analysis with radiation damping and active free field, Sect. 6.9.

Run Free Field Perform the finite element analysis of the free fields, and define the boundary conditions for the mesh, Sect. 6.10.

Write .ctrl File Save the control file, Sect. 6.11.

Generate Merlin .inp file , Sect. 6.12

6.1 Title

The user may provide a description the contents of the input file within this field. This line will be written verbatim into the beginning of the Merlin input file, Fig. 6.2.

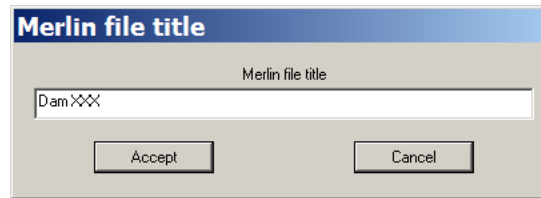


Figure 6.2: Title Data Entry

It is strongly recommended to include the selected units in the Title.

6.2 Keywords

The Keywords-menu allows the user to define general control options for the subsequent Merlin analysis, Fig. 6.3.

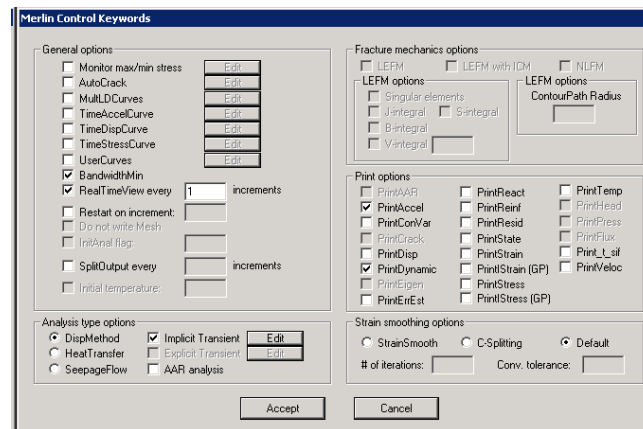


Figure 6.3: Keywords Specification

6.2.1 General options

6.2.1.1 Monitor Max/Min Stress

Specify which maximum stress component is to be monitored throughout the analysis and its maximum reported (and corresponding location) reported for each increment.

Merlin:
??

6.2.1.2 AutoCrack

This keyword is associated with the module Cracker which drives Merlin for automatic crack propagation. When Merlin senses that either a crack is to nucleate, or a crack must extend, thus requiring remeshing to accomodate discrete cracks, then control is passed back from Merlin to Cracker, Fig. 6.5.

Merlin:
1.1.11

Automatic Propagation Allows merlin to remesh if a crack wants to propagate. The user can specify the number of instances that crack stability requirements have been violated before remeshing

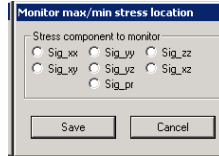


Figure 6.4: Monitor Maximum Stress

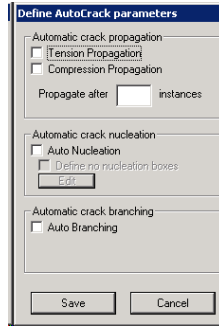


Figure 6.5: Auto-Crack Specification

takes place. Violation occurs if the stress intensity factor exceeds the fracture toughness in LEFM, or if the crack tip tensile stress exceeds the tensile strength of the material in NLFM.

Automatic Crack Nucleation Allows Merlin to interrupt the analysis if at some points the maximum tensile principal stress exceeds the tensile strength. If desired, the user can specify boxes inside which no crack nucleation can take place.

Automatic Crack Branching Will allow crack bifurcation.

6.2.1.3 MultLDCurves

This option will allow the user to track the load displacement response of a set of nodes, Fig. 6.6.

As data entry for this option can be confusing, the user should understand that:

1. One or more curves can be specified.
2. Each curve will allow the user to specify load and displacement.
3. The displacement can be either one of a specific node/dof, or the algebraic sum of multiple node/dof pairs. This is important if one needs a crack opening displacement (then one factor would be +1, and the other -1), or the displacement of a point with respect to the base. This corresponds to the Disp. #.
4. The load can be either the load at one vertex, or the entire load applied along a curve or over a patch. Again, at time one may need the resultant of a traction load.

Disp # or Load # Number of the current displacement or load monitoring point (for reference).

Entity type type of entity at which to monitor loads or displacements (vertex, curve, or patch).

LD Curve # Curve number that the current displacement or load monitoring point belongs to.

Entity # number of the current entity.

dof degree of freedom to monitor (1-2-3).

fact Factor to be applied to the displacement (note, factors are only defined for displacements).

Merlin:
1.1.10.2

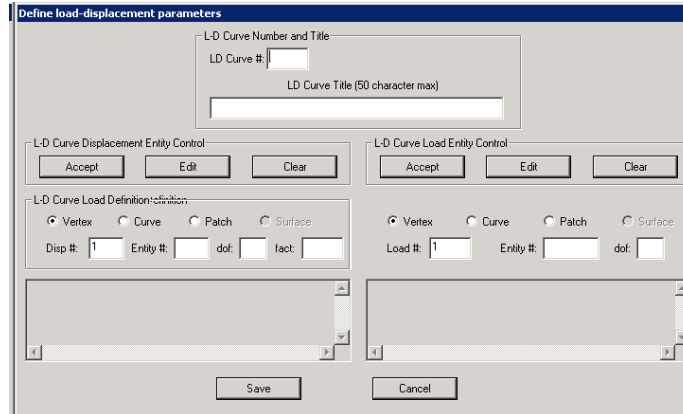


Figure 6.6: Load Displacement Curve Definition

6.2.1.4 TimeAccelCurve

A single time-acceleration curve may monitor the accelerations vs. time (or increments) at one or more vertices, 6.7:

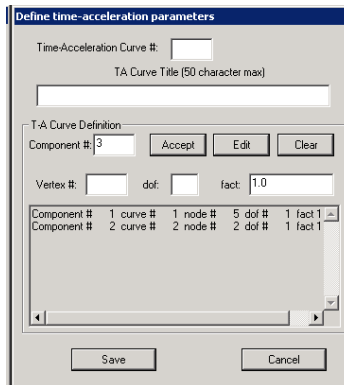


Figure 6.7: Time Acceleration Curve Definition

Case # Number of the current monitoring point (for reference).

Curve Title to identify it amongst other plots.

TA Curve # Curve number that the current monitoring point belongs to.

Vertex # Vertex number of the current monitoring point.

dof Degree of freedom to monitor (acceleration).

component Stress component to monitor.

fact Factor to be applied to the acceleration/displacement/stress at the current monitoring point.

6.2.1.5 TimeDispCurve

Definition of one or more time-displacement is performed using the Define time-displacement parameters dialogs, 6.8:

Merlin:
1.1.10.3

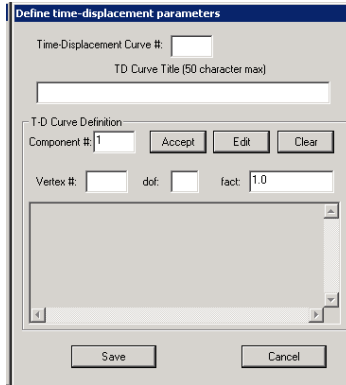


Figure 6.8: Time Displacement Curve Definition

Case # Number of the current monitoring point (for reference).

Title to be assigned to the curve.

TA Curve # Curve number that the current monitoring point belongs to.

Vertex # Vertex number of the current monitoring point.

dof Degree of freedom to monitor.

fact Factor to be applied to the displacement at the current monitoring point.

6.2.1.6 TimeStressCurve

Definition of one or more time-stress is performed using the Define time-stress parameters dialogs, 6.9:

Merlin:
1.1.10.5

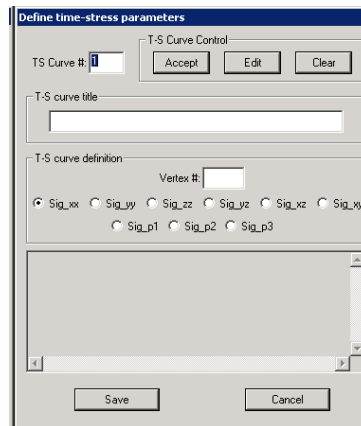


Figure 6.9: Time Stress Curve Definition

Case # Number of the current monitoring point (for reference).

TA Curve # Curve number that the current monitoring point belongs to.

Title to be assigned to the curve.

Vertex # Vertex number of the current monitoring point.

component Stress component to monitor.

fact Factor to be applied to the acceleration/displacement/stress at the current monitoring point (may be useful to convert units).

6.2.1.7 TimeStrainCurve

Definition of one or more time-strain is performed using the Define time-strain parameters dialogs, 6.10:

Figure 6.10: Time Strain Curve Definition

Case # Number of the current monitoring point (for reference).

TA Curve # Curve number that the current monitoring point belongs to.

Title to be assigned to the curve.

Vertex # Vertex number of the current monitoring point.

component Stress component to monitor.

fact Factor to be applied to the acceleration/displacement/strain at the current monitoring point (may be useful to convert units).

6.2.1.8 UserCurves

In various instances of Merlin, User may define specific x-y curves (such as for G_F , Dynamic uplift etc). This option enables the user to explicitly specify one or more curve, which will then be referenced in the input data.

6.11

Merlin:
1.1.6

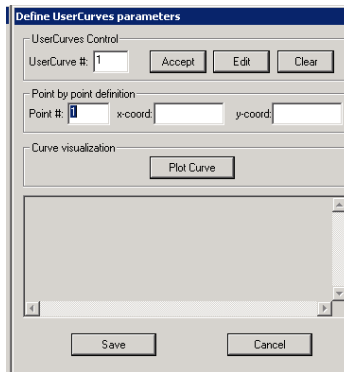


Figure 6.11: User Defined Curve

6.2.1.9 BandwidthMin

This is an optional setting that allows the user to minimize the time Merlin spends decomposing the global stiffness matrix.

6.2.1.10 RealTimeView

Requests the generation of an external file containing the displacements and accelerations at selected nodes every n increments. Merlin will then generate an `.rtv` file for real time view during the analysis by Spider.

Merlin:
1.1.12

6.2.1.11 Restart on Increment

This option enables Merlin to use the previously generated `.pst` file to restart at a certain increment. This is most useful when first a static analysis is performed, followed by a dynamic one.

Merlin:
1.1.3

6.2.1.12 Do Not Write Mesh

This option specifies that for the current Merlin input file, there is no need to write the mesh data, as those may be coming from the restart file.

6.2.1.13 InitAnalflag

Missing

6.2.1.14 Split Output

In some analysis, the ASCII output file may be too large for the operating system or for the editor. This option enables the user to split the output into multiple files, each containing the results of n increments.

Merlin:
1.7.21

6.2.1.15 Initial Temperature

This is the stress free initial temperature, needed only for AAR analysis.

Merlin:
1.1.9

6.2.2 Analysis Type Options

DispMethod tells the code that the analysis will be traditional, structural, finite element analysis.

Heat Transfer allows the user to specify that a heat transfer analysis be to be performed. Selection of this option will preclude selection of analysis options relating to stress or seepage problems.

Merlin:
1.3.6

SeepageFlow alerts the computer to an impending seepage flow analysis problem.

Thermo-elast option not yet available.

Poro-elast option not yet available.

6.2.2.1 Implicit Transient

For transient analysis, user must specify, Fig. 6.12:

Merlin:
1.3.4

Integration Scheme Newmark's β method or Hughes α method.

delta T corresponding to the ΔT of the transient analysis.

Hughes alpha coefficient.

Accelerations models during the time step (to automatically determine Newmark's coefficients).

Newmark beta β coefficient for Newmark's time integration (only if user defined). Note that KumoNoSu predefines β as 1/4 and 1/6 for constant and linear acceleration over the time step.

Newmark gamma γ coefficient for Newmark's time integration (only if user defined). Note that KumoNoSu predefines β as 1/2 and 1/2 for constant and linear acceleration over the time step.

Figure 6.12: Data Entry for Transient Analysis

Nodal Acceleration Definition check if accelerations are defined in each increment or if the accelerations are prescribed in a single block.

Stiffness damping stiffness damping parameter α for Rayleigh damping

Mass damping mass damping parameter β for Rayleigh damping

Compute Rayleigh Damping coefficients. to facilitate determination of the Rayleigh damping coefficients, user can simply specify the two frequencies f [Hz] and corresponding damping [%], KumoNoSu will then compute the coefficients, and plot them Fig. 6.13.

OrthoMass Parameters if an earthquake is not aligned with a principal axis, user can use **orthomass** to define an orthotropic mass matrix. the two (or three) γ coefficients, each less than 1., premultiply the mass matrix along that direction.

6.2.2.2 Explicit Transient

Initial time step time increment which should be very carefully determined to ensure stability.

Coefficient for critical timestep see [Merlin Manual](#).

Stiffness and Mass Damping for Rayleigh damping.

Nodal Acceleration Definition check if accelerations are defined in each increment or if the accelerations are prescribed in a single block.

6.2.3 Strain Smoothing Options

StrainSmooth instructs the computer to 'improve' the values for nodal strain through iterations using the technique of Zienkiewicz, Kui, and Nakazawa. No inversion of the consistent projection matrix is required (thus, the smoothing procedure is computationally inexpensive).

Merlin:
1.3.5

Merlin:
3.4.1.10.2

Merlin:
1.5

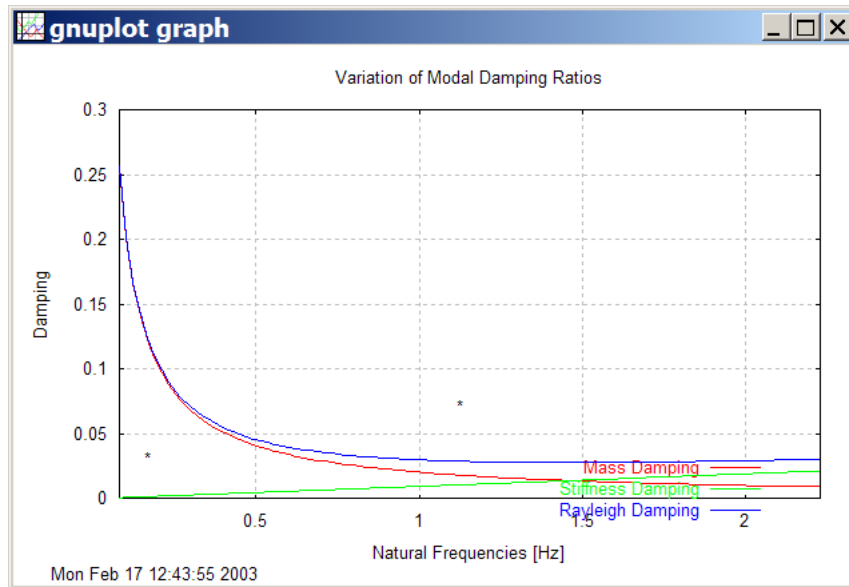


Figure 6.13: Computed Rayleigh Damping Coefficients

Define explicit transient analysis parameters

Explicit transient analysis parameters

Initial timestep: 0.1

Coefficient for critical timestep: 0.1

Stiffness damping: 0.0

Mass damping: 0.0

Nodal acceleration history input

Accelerations incrementally specified

Accelerations prescribed in one block

Save Cancel

Figure 6.14: Data Entry for Explicit Transient Analysis

C-Splitting instructs the computer to 'improve' the values for nodal strain through iterations. Very similar to StrainSmooth, but C-Splitting converges faster.

Default instructs the computer to calculate strain with a lumped projection matrix.

6.2.4 Fracture Mechanics Options

LEFM sets a linear elastic fracture mechanics analysis for use in stress analysis only.

Merlin:
1.4

J-Integral specifies the J-integral (Rice, Hellen, and Blackburn) to be used to indirectly calculate stress intensity factors from energy quantities.

S-Integral specifies the S-integral (Stern and Hong) to directly compute the stress intensity factors based on reciprocal work.

B-Integral specifies the B-integral (Babuska and Miller) for calculation of stress intensity factors based on energy release.

V-Integral Specifies that the volume integral (based on J) is to be used to determine the SIF in 3D.

ContourPath Radius defines the radius of the contour around the crack tip (2D) or the crack front (3D) from which the J/B or V integrals will be determined.

NLFM For nonlinear fracture mechanics (based on the cohesive crack model which requires the insertion of icm elements along the crack.

6.2.5 Print Options

PrintAAR prints incremental AAR strains.

Merlin:
1.7

PrintAccel prints nodal accelerations at each increment.

PrintConVar prints nodal constraint variables from stress analysis to output file.

PrintCrack prints crack displacement, pressure, and stress profiles from fracture mechanics.

PrintDisp user to specify the nodal displacements for stress analyses are printed to the output file.

PrintDynamic prints in an external file results of dynamic analysis, [check](#).

PrintEigen prints results of the eigenvalue analysis.

PrintErrEst prints error estimates for strain energy and strain resulting from a stress analysis.

PrintReact user specifies the nodal reactions for a stress, heat flow or seepage flow analysis are printed to the output file.

PrintReinf prints embedded reinforcement information to the output file.

PrintResid prints residuals from stress, heat or seepage analysis to output file.

PrintState prints state variables from stress analysis to output file.

PrintStrain prints nodal strains from a stress analysis to the output file.

PrintStrain (GP) allows user to print Gauss points strains from a stress analysis to the output file.

PrintStress prints nodal stresses from a stress analysis to the output file.

PrintStress (GP) allows user to print Gauss points stresses from a stress analysis to the output file.

PrintTemp prints nodal temperatures to output file following heat transfer analysis.

PrintHead prints nodal heads for a seepage flow problem to output file.

PrintPress prints nodal heads in form of pressures to output file following seepage analysis.

PrintFlux prints nodal fluxes to output file following heat transfer or seepage flow analysis.

Print_t_sif prints mode I stress intensity factors for each time increment of dynamic crack propagation analysis. This option must be used together with Transient, Impact, and LEFM.

PrintVeloc prints nodal velocities.

6.3 Material and Element Groups

KumoNoSu keeps track of: a) the number of material and element which must be defined, and b) if the problem is 2D or 3D. Hence, when the Material tag is selected, the user is prompted to identify those, Fig. 6.15.

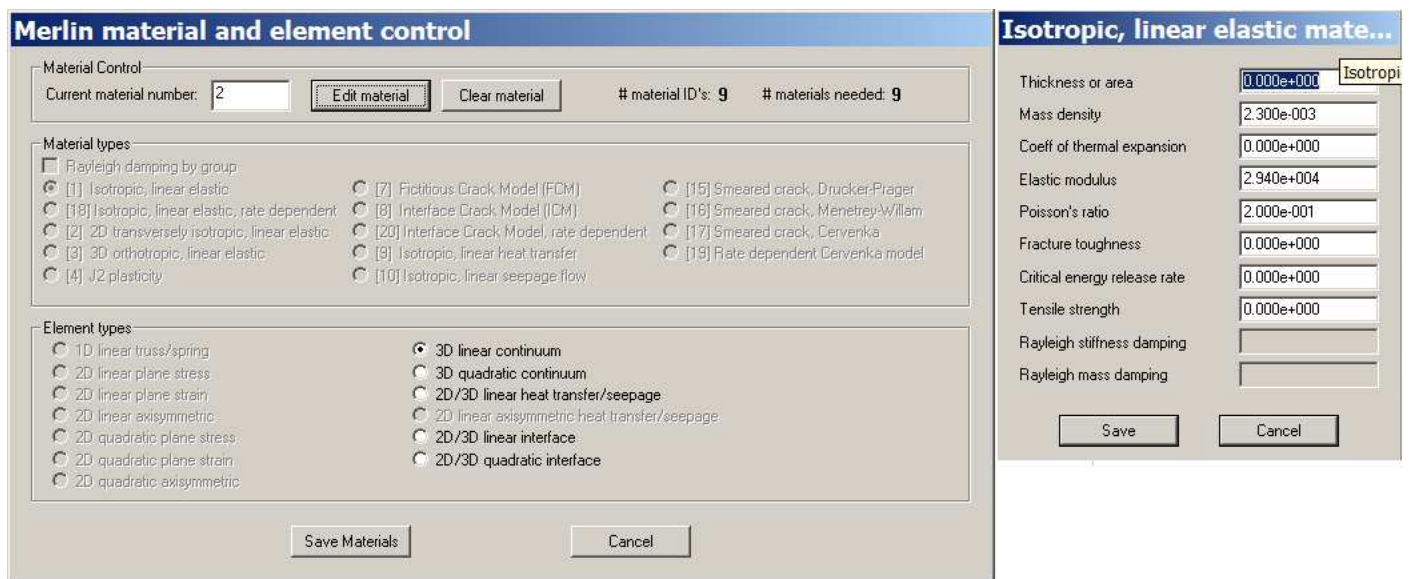


Figure 6.15: Material Input Data

Element Type This is done by indicating simply the type of element (only those possible in the context of the current analysis are enabled, the others are greyed out).

The corresponding Constitutive Model. Again, only those possible are enabled.

Once a material type has been selected, KumoNoSu will then pop-up a second window inside which the user will enter the corresponding material properties. Whereas more details about these can be found in the Merlin manual, the sections below describe the data to be specified for each material type.

6.4 AAR

6.5 Discrete/Continuum Groups

6.6 Eigenmode

Eigenvalue analysis is also possible, Fig. 6.18. There are two choices:

Merlin:
2.2.2
Merlin:
2.2.3

Merlin:
2.2.4

Merlin:
2.2.3.5

Merlin:
3.4.1.9.1

Figure 6.16: AAR User Interface

Figure 6.17: Discrete ContinuumUser Interface

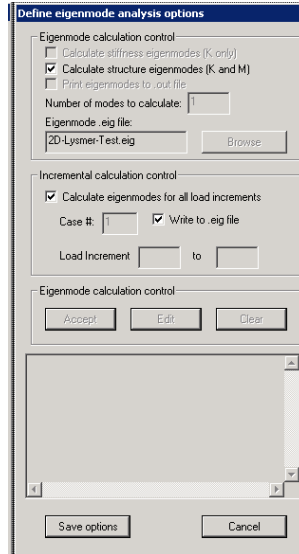


Figure 6.18: Eigenmode Specification

1. Natural frequencies and mode shapes of the structures determined from

$$(\mathbf{K} - \lambda\mathbf{M})\mathbf{u} = 0 \quad (6.1)$$

2. Eigenmodes (and mode shapes) of the stiffness matrix only.

$$(\mathbf{K} - \lambda\mathbf{I})\mathbf{u} = 0 \quad (6.2)$$

this is an important feature when a nonlinear analysis is performed, as it enables the analyst to check the number of zero (or the decrease of the lowest ones) eigenvalues as damage progresses inside the structure. Eigenmodes can be computed either for all increments or for selected ones.

In both cases, the user must specify the **number of eigenmodes** to be determined (the larger the number, the longer the CPU time)

Results can be printed on a separate file, selected by **browse**. This file in turn can be viewed by **Spider** which will display and animate the mode shapes.

6.7 Loads

The load group is divided into two parts: Incremental and Total. The Incremental group of loads is a load applied only for the specified increments. On the other hand the Total load, once applied remains in effect for all subsequent increments.

A note on

6.7.1 Incremental Load Definition

6.7.1.1 Displacement BC's

Merlin considers the boundary conditions as an intrinsic part of the load. Boundary conditions can be specified for individual vertices, curve, patch or surface, Fig. 6.20. The following must be specified:

Load increment(s) Define load increment or increment interval for current bc. This enables the user to specify the same boundary conditions for some or all the increments.

Merlin:
3.4.1.3

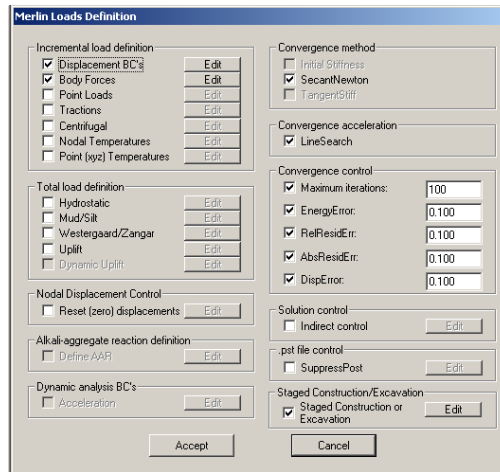


Figure 6.19: Loads Type

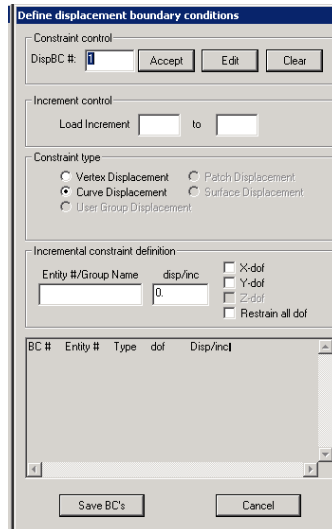


Figure 6.20:

Constraint type Entity type to be restrained (vertex or curve in 2D; vertex, curve, patch, or surface in 3D). If the selected entity type has minor entities associated to it (such as the end vertices for a curve, or curves to a surface/patch), those minor entities may be restrained also by selecting **Restrain associated minor entities**.

DispBC # Number of the current bc (for reference only).

Entity # Number of the entity to be restrained

DOF # Degree of freedom to be restrained If every dof of the current entity is to be restrained, select Restrain all dof to do so

disp/inc Displacement magnitude per increment (usually zero which is the default value).

6.7.1.2 Body Forces

The body forces are defined by specifying the acceleration of gravity to be applied to all or selected material group. Each group in turn would have had its mass defined separately. Hence, we can have a material with a mass (thus subjected to an acceleration force), but without gravity load (such as the rock foundation in a dynamic analysis of a dam), Fig. 6.21.

Merlin:
3.4.1.8

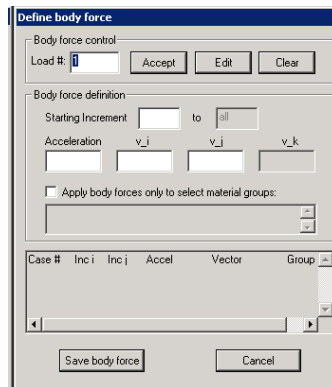


Figure 6.21: Definition of Body Forces

Load # Number of the current load (for reference).

Load increment(s) Two possible cases:

1. If all materials are subjected to body force, only specify the increment in which body forces are first required. If only select materials are subjected to body force, specify the beginning and end load increments for this material group list.

Acceleration Magnitude of the acceleration applied to the materials.

$v_i, v_j, (v_k)$ Components of the vector defining the direction. of acceleration.

Select groups If option 2 is required, check the box and defined the select material group numbers that are subjected to body forces. Careful this is for material group and not regions or patches.

6.7.1.3 Point Loads

Fig. 6.22.

Merlin:
3.4.1.4

Load increment(s) Define load increment or increment interval for current load. Remember that this is an *incremental* load.

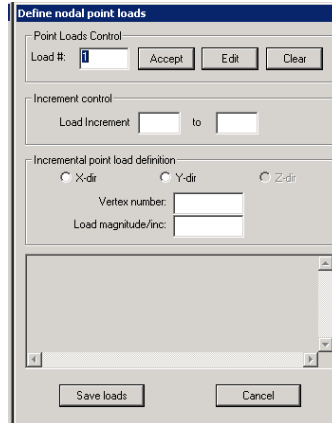


Figure 6.22: Point Load Definition

Load number Number of the current load (for reference).

Vertex number Number of the vertex to be loaded.

Load magnitude/inc Magnitude of the point load per load increment (ΔP).

Load direction X-dir, Y-dir, or Z-dir (3D only)

6.7.1.4 Tractions

Traction loads on curves, patches or surfaces is defined as follows, Fig. 6.23:

Merlin:
3.4.1.5

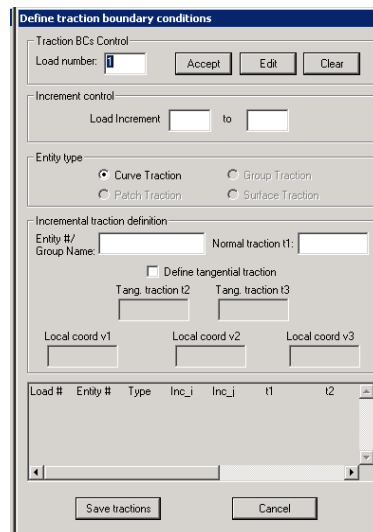


Figure 6.23: Traction Load Definition

Load number number of the current load (for reference).

Load increment(s) Define load increment or increment interval for current load.

Entity type entity type to be loaded (2D: curve only; 3D: curve, patch, or surface).

Entity number number of loaded entity

Normal traction t_n traction load applied normal to the element surface (positive = tension, negative = compression).

Tang. Traction t_{n1} , t_{n2} tangential traction components in the local y -axis (t_{n1}) and z -axis (t_{n2}); t_{n2} traction only active for 3D models

Local coordinates v_1 , v_2 , v_3 Vector defining the local coordinate system for the element surface; only required for 3D models, Fig. 6.24.

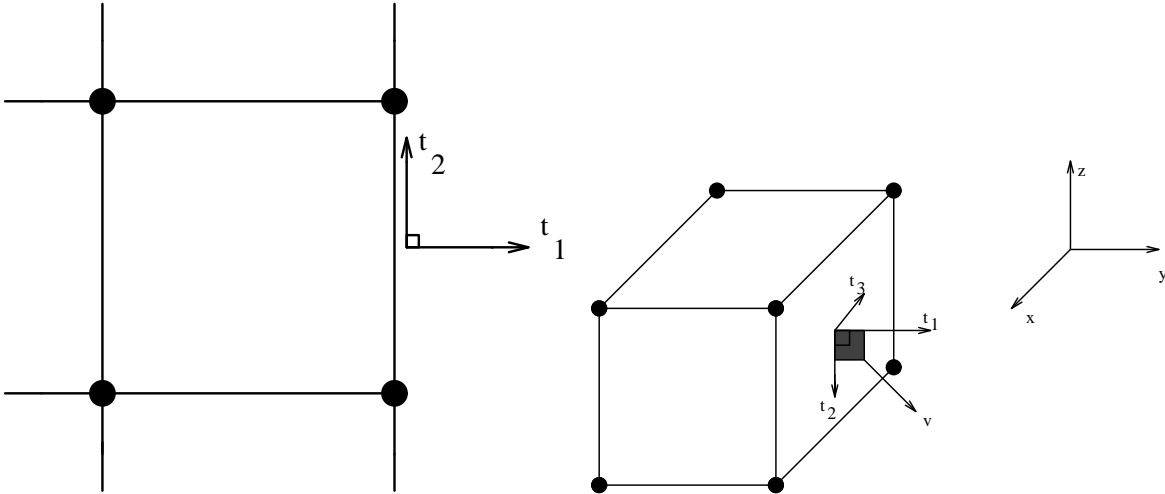


Figure 6.24: Direction of Traction load

Note that for 2D problems, the traction is defined for unit thickness, hence Merlin will internally multiply the thickness t .

6.7.1.5 Centrifugal

The **Centrifugal** option enables the user to define the incremental centrifugal forces for stress analyses. This option is to be used when the centrifugal (or gravity) forces varies along a structure (such as in tests performed inside a centrifuge, or electromagnetic forces), Fig. 6.25, and the force is given by

$$F = m\omega^2 r \quad (6.3)$$

where ω is the angular velocity (rad/s).

Load number Number of the current load (for reference).

Load increment(s) Define load increment or increment interval for current load.

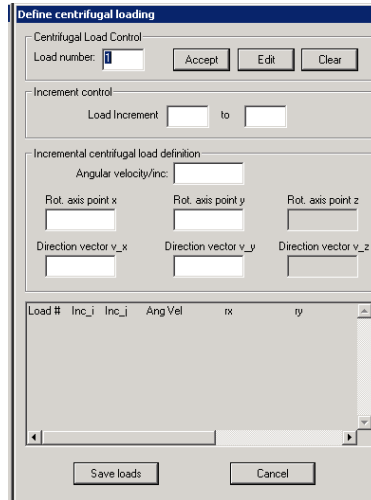
Angular velocity Magnitude of angular velocity ω per load increment.

Axis point x_i , x_j , x_k Coordinates of a point on the axis of rotation.

Direction vector v_i , v_j , v_k Vector specifying the direction of rotation.

Note: once defined, centrifugal loading is applied to the entire model.

Merlin:
3.4.1.11



Define centrifugal loading

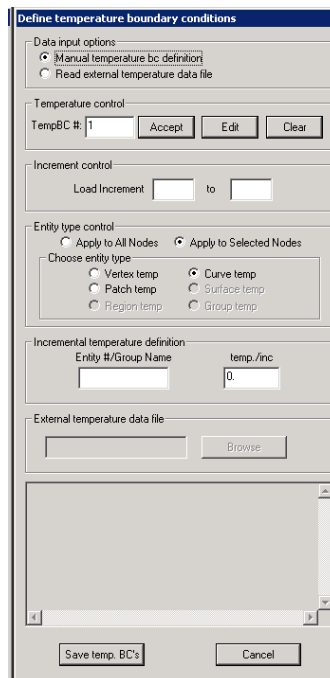
Centrifugal Load Control
 Load number:

Increment control
 Load Increment to

Incremental centrifugal load definition
 Angular velocity/inc:
 Rot. axis point x Rot. axis point y Rot. axis point z
 Direction vector v_x Direction vector v_y Direction vector v_z

Load #	Inc_j	Inc_i	Ang Vel	rx	ry

Figure 6.25: Centrifugal Load Definition



Define temperature boundary conditions

Data input options
 Manual temperature bc definition
 Read external temperature data file

Temperature control
 TempBC #:

Increment control
 Load Increment to

Entity type control
 Apply to All Nodes Apply to Selected Nodes
 Choose entity type
 Vertex temp Curve temp
 Patch temp Surface temp
 Region temp Group temp

Incremental temperature definition
 Entity #/Group Name temp./inc

External temperature data file

Figure 6.26: Nodal Temperature Load Definition

Figure 6.27: Point Temperature Load Definition

6.7.1.6 Nodal Temperature

6.7.1.7 Point (xyz) Temperatures

Merlin:
3.4.1.13.1
Merlin:
3.4.1.13.2

6.7.2 Total Load Definition

Total loads are those which once applied, remain constant in all subsequent load increments.

6.7.2.1 Hydrostatic

Hydrostatic load is defined by Fig. 6.28.

Merlin:
3.4.1.7.1

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (curve in 2D, patch or surface in 3D).

Load number Number of the current load (for reference).

Entity # Number of the loaded entity.

Elevation/inc Fluid elevation change per load increment (Δh)

Fluid weight Fluid weight density (force/volume).

Direction of fluid height in global coordinates, X-dir, Y-dir, or Z-dir (3D only).

Note that in general, a first load number must be defined which brings the water elevation to the base of the dam. Then, incremental water height (Δh) can be defined to raise the water at each increment.

6.7.2.2 Mud/Silt

Mud and silt load is slightly different than hydrostatic load, as the lateral pressure is equal to the vertical pressure times the passive coefficient k . For an inclined curve (2D) or patch (3D), KumoNoSu

Figure 6.28: Hydrostatic Load Definition

will automatically calculate the mud/silt pressure normal to the surface, given the vertical and lateral components F_a and F_l .

It is defined, as follows, Fig. 6.29.:

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (curve in 2D, patch or surface in 3D).

Load number Number of the current load (for reference).

Entity # Number of the loaded entity.

Elevation/inc Fluid elevation change per load increment (Δh).

Fluid weight Fluid weight density (force/volume).

Passive pressure coeff. Passive pressure coefficient for lateral pressure due to mud/silt (k coefficient).

Direction of fluid height in global coordinates, X-dir, Y-dir, or Z-dir (3D only).

As for hydrostatic load, a first load number must be defined which brings the mud elevation to the base of the dam. Then, incremental mud height (Δh) can be defined to raise the mud at each increment.

6.7.2.3 Westergaard

Westergaard's pseudo-hydrodynamic added masses are determined as follows, Fig. 6.30:

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (curve in 2D, patch or surface in 3D).

Load # Number of current load (for reference).

Entity # number of the loaded entity.

Merlin:
3.4.1.7.3
3.4.1.7.3

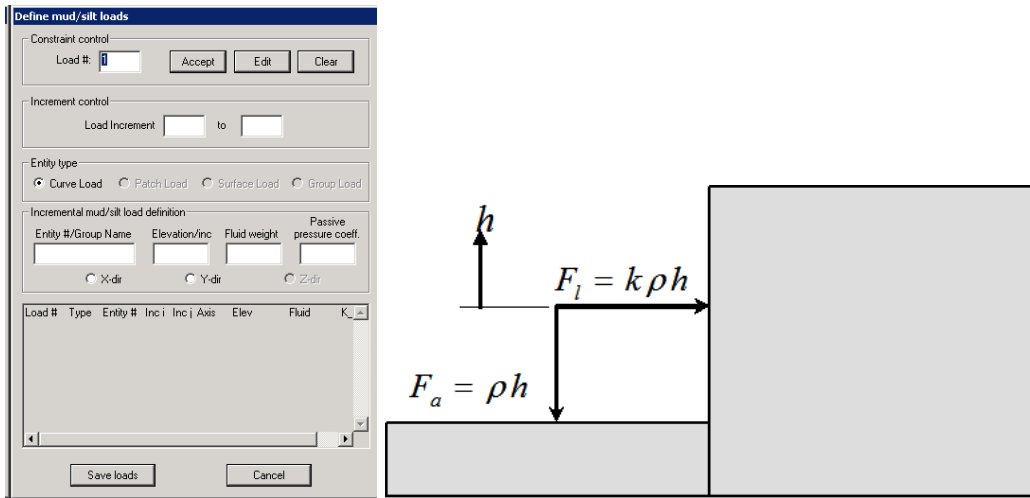


Figure 6.29: Mud Silt Definition

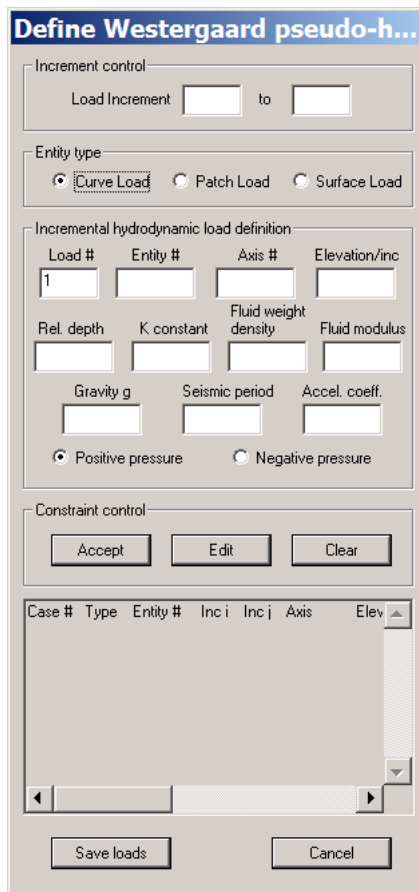


Figure 6.30: Westergaards Added Mass Load Definition

Axis # Axis defining reservoir depth: 1 = X, 2 = Y, 3 = Z (3D only).

Elevation/inc Elevation of the reservoir surface per load increment note that this is not the relative depth of the reservoir, but rather the elevation of the surface).

Rel. depth Relative depth of the reservoir.

K constant : K constant for Westergaard equation (defined by Westergaard as 51.0 lb/ft³ or 8,011.4 N/m³ for water).

Fluid weight Weight per unit volume of the fluid.

Fluid modulus Elastic modulus of the fluid (may be taken as 300 kips/in² or 2.068 GPa for water).

Gravity g Acceleration of gravity.

Seismic period Period of the earthquake motion.

Accel coeff Lateral acceleration as a fraction of gravity.

Pressure type Positive (towards dam) or negative.

Note that if the earthquake excitation is not along the stream axis, than an orthogonal model of the added mass must be specified, Fig. 6.31:

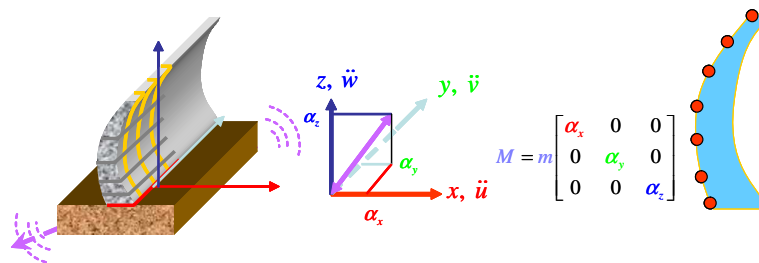


Figure 6.31: Westergaard's Orthogonal Added Mass Load Definition

6.7.2.4 Uplift

Two uplift models are implemented in KumoNoSu :

Full is when we have an uplift pressure, only along a crack, equal to the upstream hydrostatic pressure. There is no uplift pressure from the crack tip to the downstream face.

Ferc is identical to Full, however there is a linearly varying pressure from the crack tip to the tailwater hydrostatic pressure, Fig. 6.32.

KumoNoSu defines the uplift as follows, Fig. 6.33:

Load increment(s) Define load increment or increment interval for current load.

Uplift model type Full uplift or the FERC model.

Load # Number of the current load (for reference).

Crack Path # Number of the crack path subjected to uplift pressures.

Axis # Axis defining reservoir depth: 1 = X, 2 = Y, 3 = Z (3D only).

Elevation/inc Fluid elevation per load increment.

Merlin:
3.4.1.7.4

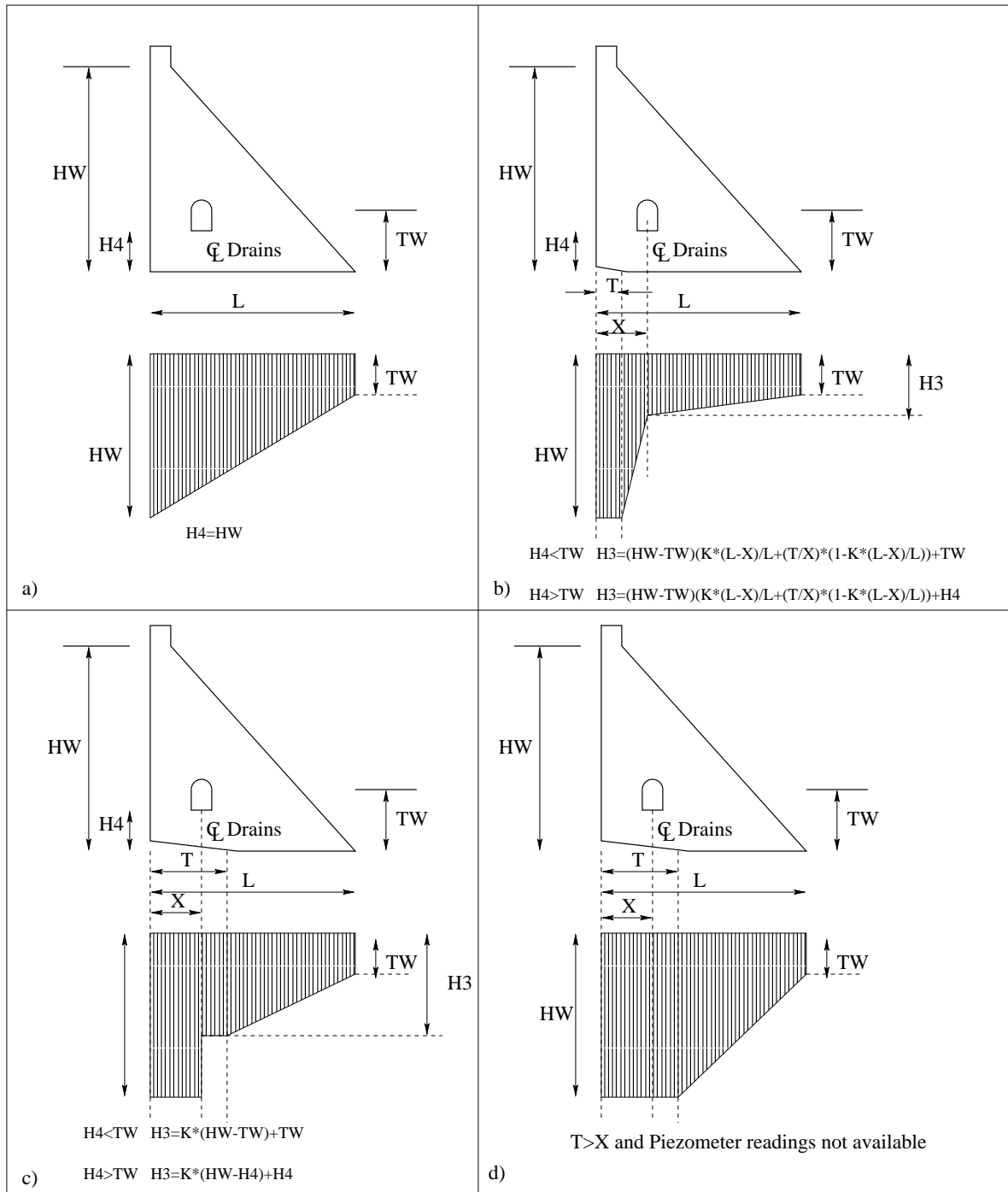


Figure 6.32: FERC Uplift Loads

Figure 6.33: Uplift Load Definition

Fluid weight Weight per unit volume of the fluid.

Crack prop axis # Global axis which defines the direction of crack propagation (necessary for 3D uplift only).

Crack surfaces subjected to uplift : Both sides, lower surface only, or upper surface only.

If the FERC model is used, Fig. 6.32, then additional data must be entered:

Distance from crack mouth to the drain This is an average distance.

Drain efficiency e ($0 < e < 1$) where 1. corresponds to full efficiency)

- Before it is intersected by the crack. If there is no drain, specify 0.
- After it has been intersected by the crack.

Drain elevation Vertical distance between the crack subjected to uplift and the drain

Tail water elevation (or water elevation at the end of the crack).

Total base length of the dam, i.e from upstream to downstream. Note this is not the projected length, but the actual length of the crack. For multiple (inclined) segments, the total “curvilinear” length must be specified. If the base length is variable, then it would be more conservative to adopt the maximum length.

6.7.2.5 Dynamic Uplift

to insert

Merlin:
3.4.1.7.5

6.7.3 Reset Nodal Displacements

This enables the user to rest the nodal displacements (but not the stresses or the state variables) to zero after a certain increment, Fig. 6.34.

Merlin:
3.1.2

6.7.4 Heat Transfer/Seepage Loads

6.7.4.1 Temperature

Temperature nodal loads can be specified not only for thermal effects but also for any other physical phenomena causing expansion (such as Alkali-Silica Reactions, in conjunction with a pseudo coefficient of thermal expansion defined in the material properties), Fig. 6.35.

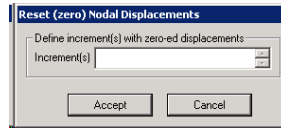


Figure 6.34: Reset (zero) Displacements Definition

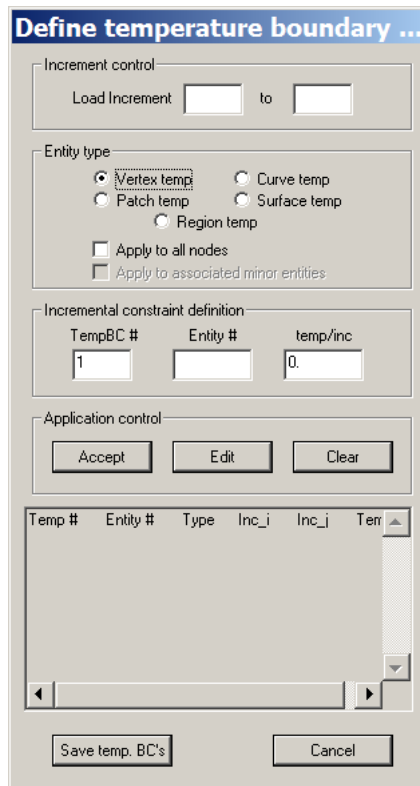


Figure 6.35: Temperature Load Definition

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (vertex or curve in 2D, all entity types in 3D). Note that:

- If every node in the FE mesh is to receive a temperature bc, select **Apply to all nodes**.
- If the selected entity type has minor entities associated to it (such as the end vertices for a curve), those minor entities may be included by selecting **Apply to associated minor entities**.

TempBC # Number of current bc (for reference).

Entity # Number of the entity receiving temperature bcs.

Temp/inc Temperature change (ΔT) per load increment.

6.7.4.2 Head

Head load is defined as follows, Fig. 6.36.

The dialog box is titled "Define fluid head boundary co...". It features the following sections and controls:

- Incremental constraint definition:** A text box containing "Define fluid head boundary conditions" and two input fields for "Load Increment" and "to".
- Application type:** Two radio buttons, "Nodal Head" (selected) and "Curve Head".
- Incremental constraint definition:** Three input fields: "Head #" (value 1), "Vertex/Curve #" (empty), and "head/inc" (value 0).
- Application control:** Three buttons: "Accept", "Edit", and "Clear".
- Table:** A table with columns: "Head #", "Head type", "V/C #", "Inc i", "Inc j", and "Head". The table is currently empty.
- Buttons:** "Save Heads" and "Cancel" buttons at the bottom.

Figure 6.36: Head Load Definition

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (vertex or curve).

Head # Number of current head(for reference).

Entity # Number of the entity receiving the head bcs.

Head/inc Head per load increment.

6.7.5 Dynamic Analysis

This option is intended to be used for the application of an acceleration history for a dynamic analysis. Since the history is applied at load increment 1 of the dynamic analysis, no load increments or intervals may be specified, Fig. 6.37. Mandatory information:

Figure 6.37: Acceleration Specification

Acceleration history file Name of text file containing the nodal acceleration history. This file should contain: $t_i, a_{xi}, a_{yi}, a_{zi}$.

.pst file data dump Increment interval at which to write analysis results to the .pst output file for viewing with Spider.

Constraint type Entity type receiving acceleration bcs (vertex or curve in 2D, vertex, curve, or patch in 3D).

BC # Number of current acceleration bc (for reference).

Entity # Number of the entity to receive acceleration bcs.

DOF Degree of freedom to receive acceleration bcs.

6.7.5.1 Harmonic Excitation

In some applications, it is desirable to generate a harmonic excitation curve and apply it to the structure. If this option is selected, then the following additional data must be specified:

a_{max} Maximum acceleration magnitude.

frequency Frequency of the harmonic motion (Hz).

Δt Time step for the harmonic motion (may be overwritten by KumoNoSu if too large to capture the harmonic motion).

total t Total time of harmonic motion.

6.7.6 Convergence Control

6.7.6.1 Convergence Control

Convergence control:

Maximum Iterations specify maximum number of iterations

Energy/Error specify relative convergence tolerance in terms of energy (recommended value product of displacement error and the absolute residual error).

Merlin:
3.2.2

RelResidErr specify relative convergence tolerance in terms of residual forces (=10% , typically).

AbsResidErr specify absolute convergence tolerance in terms of residual forces. Especially useful for stress analysis involving thermal gradients (recommended value: 10%).

Merlin:
3.2.3
Merlin:
3.2.5

DispError specify relative convergence tolerance in terms of displacements. (recommended value: 1%).

6.7.6.2 Solution Method

KumoNoSu enables the user to select any one of the following:

1. Initial Stiffness (which is the default mode for Merlin.
2. Secant Newton.
3. Tangent Stiffness.

Merlin:
3.3

and Line Search may also be specified.

Merlin:
3.3.1.2
Merlin:
3.3.1.1

6.7.6.3 Convergence Acceleration

6.7.7 Solution Control

If an indirect solution control (such as crack opening displacement) is selected, Fig. 6.38. The **SpecifyCOD** option enables the user to control the step size for the modified-Newton algorithm by specifying a relative displacement component between two nodes (de Borst 1986, de Borst 1987).

Merlin:
3.3.2
Merlin:
3.3.3

Load Increment from first and last load increment.

COD Magnitude The magnitude of the prescribed relative displacement component, $\Delta u'$. Note that this is not necessarily a crack opening displacement.

Reference vertex # reference node.

Moving vertex # second node.

vector components The three components v_i of the vector \mathbf{v} defining the direction of the relative displacement component.

The magnitude of the prescribed relative displacement component and the direction vector components are floating point numbers and the node numbers are integers. The relative displacements (in the global coordinate system) are defined by subtracting the displacements for the “moving” node from the displacements for the reference node. The direction vector \mathbf{v} is not required to be a unit vector; the program utilizes it before computing the relative displacement component. For two-dimensional analyses the third component v_z of the vector \mathbf{v} is not input; it is only input for three-dimensional analyses.

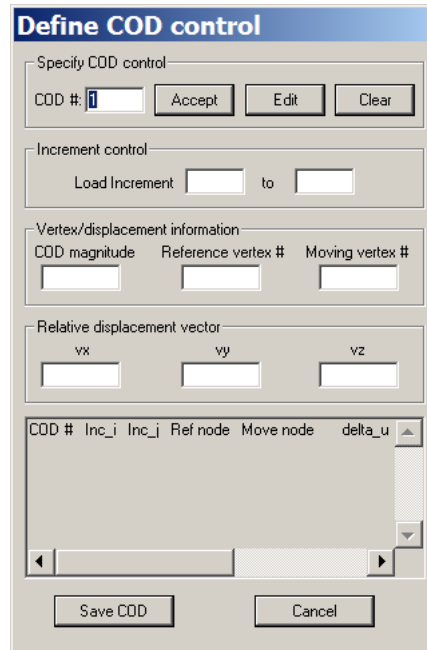


Figure 6.38: Indirect (COD) Solution Control

6.7.8 .pst File Control

This option, Fig. 6.39 enables the user to minimize the size of the .pst file (used both for restart, and for the graphical postprocessor Spider. User can specify:

Merlin:
3.1.3

Case # Case number (for reference).

Load increment Specifies the range of increments inside which the .pst file is to be written at fixed interval (different from the default of 1).

Write data to .pst file every Interval of .pst file update.

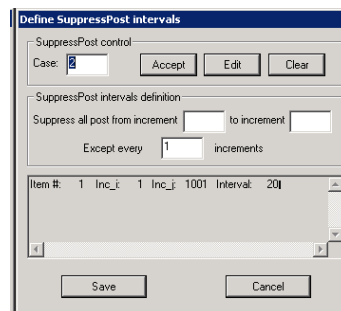


Figure 6.39: Suppress pst Output

6.7.9 Staged Construction/Excavation

Staged construction/excavation can be approximately simulated by the algorithms shown in Fig. 6.40 and 6.41 respectively.

Though clearly an approximation, this approach was relatively simple to implement, and yields excellent results.



Figure 6.40: Algorithm for Staged Construction

1. With reference to Fig. 6.42, the user must begin by assigning the exact material properties for **Construction** all the materials, or only the reference material id. Kumo will assign an initial scaled value of E corresponding to the user entered value to the other groups.
Excavation Assign the exact material properties to all the elements.
2. Specify the reference material group id (the one with the exact properties).
3. Specify the scaling factor for E corresponding to the "ghost" elements.
4. Specify the order for construction/excavation.

It should be noted that 2 increments are needed for each additional layer of excavation or construction. Hence if not enough increments have been specified, kumo will issue a warning message, Fig. 6.43.

6.8 Incremental Material Update

This option enables the user to modify material properties after the first increment. Hence, during a load increment, user can alter any of the existing material properties, Fig. 6.44.

This option is useful to alter the material properties from the static ones, to dynamic ones after a restart. This can also be used to simulate stage construction by adding the mass and increasing the Young's modulus incrementally.

6.9 Generate Free Field

Fig. 6.45...

Merlin-T:
5.3.2

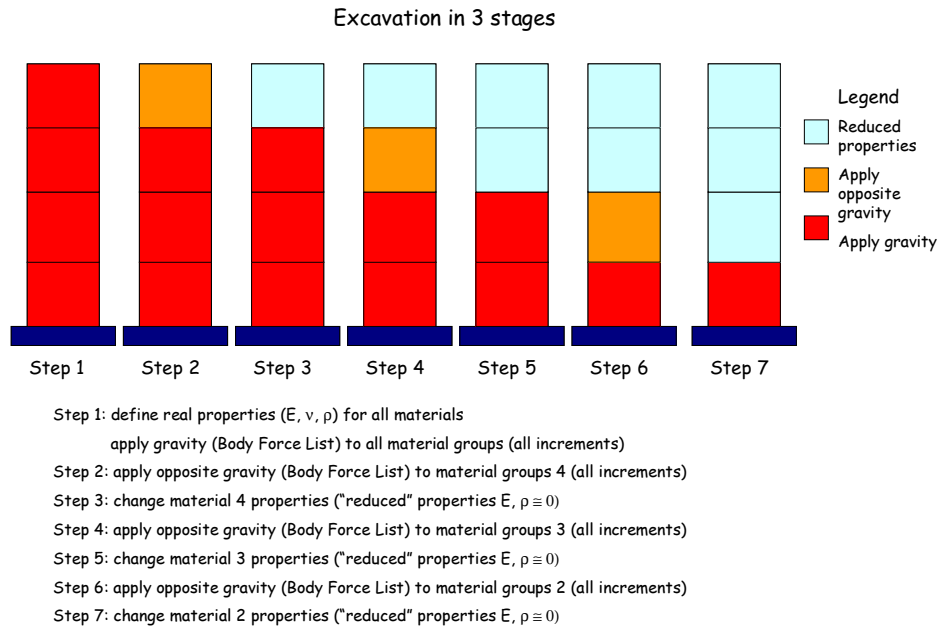


Figure 6.41: Algorithm for Staged Excavation

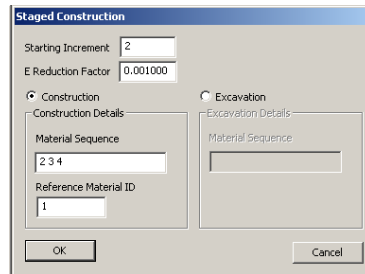


Figure 6.42: Graphical User Interface for Staged Construction/Excavation

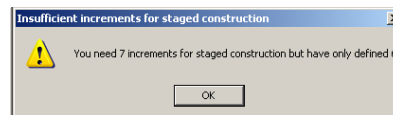


Figure 6.43: Staged COnstruction/Excavation Warning if Insufficient number of Increments Specifiedn

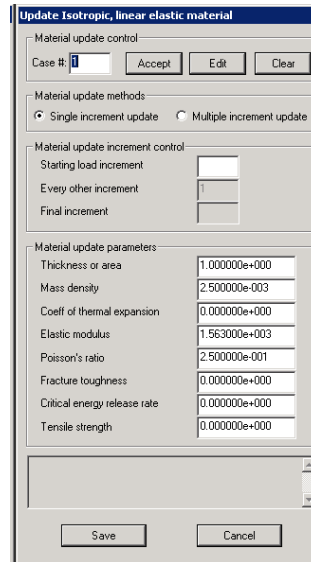


Figure 6.44: Incremental material Update

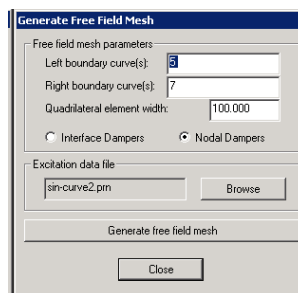


Figure 6.45: Generate Free Field

6.10 Run Free Field

This will simply run all the free field analysis, and then transfer the appropriate results to the merlin input data file of the main structure.

6.11 Write .ctrl file

A `.ctrl` file is needed to run the T3D2Merlin module. It should be noted that the Merlin input data file will be assigned the same filename as the one of the `.ctrl` file.

6.12 Generate Merlin .inp file

With all preliminary steps completed, the user may generate the *.inp-file for Merlin's analysis. The *.inp-file weds the geometric data (*.t3d-file) to the material properties and load data (*.ctrl-file). Fig. 6.45... Note that if a parallel explicit analysis is to be performed, user should specify the number of

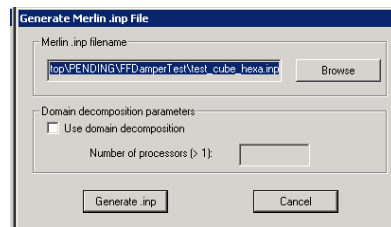


Figure 6.46: Generate Merlin Input File

processor, and an internal domain decomposition will be performed.

Chapter 7

Merlin Files

MerlinToolBox: Windows based driver for the following three programs.

KumoNoSu: Mesh Generator.

Merlin: Finite Element analysis program.

Spider: Graphical postprocessor

MATLAB: Based Drivers for Parameter Identification through Least Square Minimization

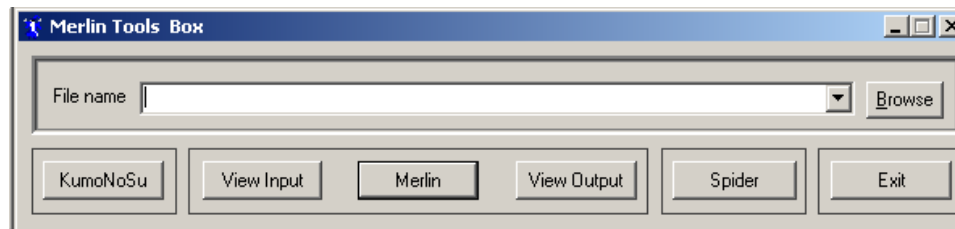


Figure 7.1: Merlin Toolbox

Extension	Used by	Generated by
Merlin Specific Files		
.inp	Merlin input file	KumoNoSu/T3D2Merlin
.out, .dyn	Merlin output text files	Merlin
.pst, .eig, .rtv	Spider input file	Merlin
KumoNoSu Specific Files		
.bd	T3D input file	KumoNoSu
.t3d	T3D output file	T3D
.ctrl	T3D2Merlin input file	KumoNoSu
Beaver Specific Files		
.dam	Beaver	Beaver
.bd	KumoNoSu	Beaver
.ctrl	KumoNoSu	Beaver

Table 7.1: File Types

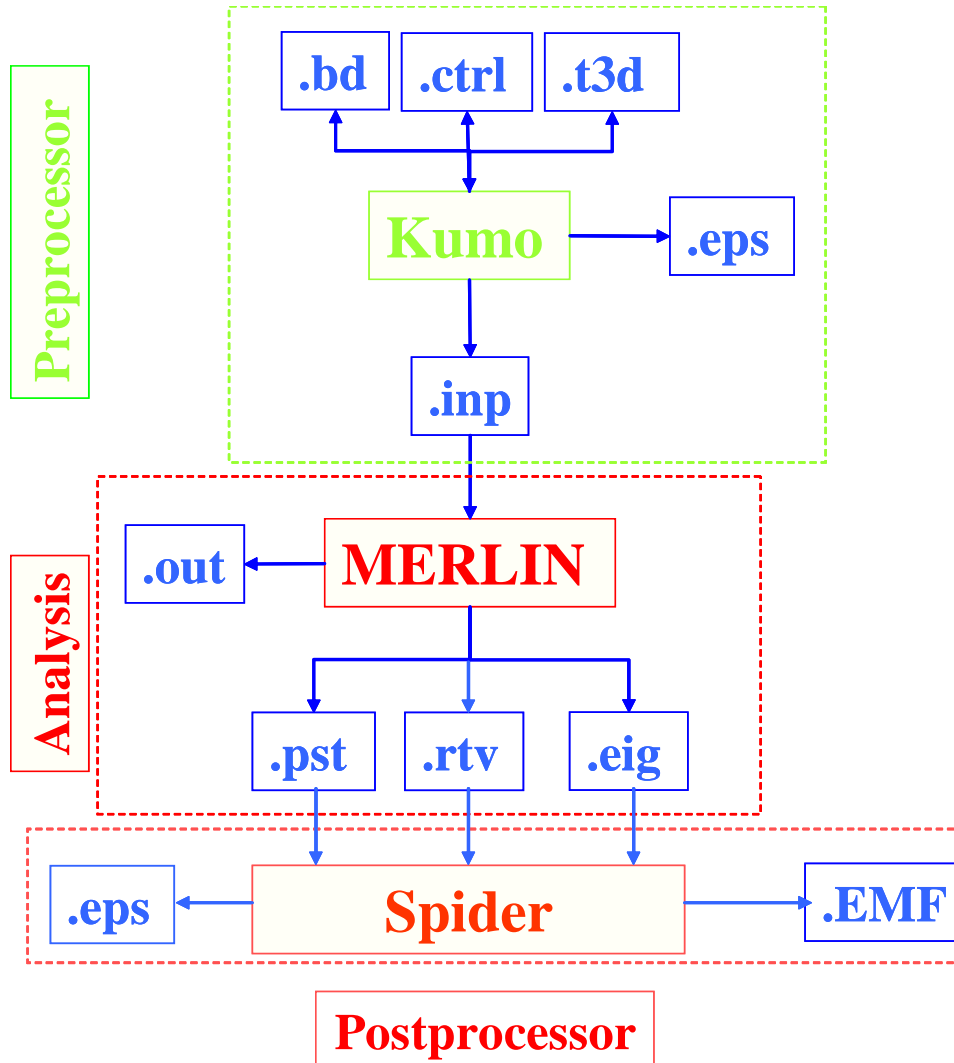


Figure 7.2: Program Interactions

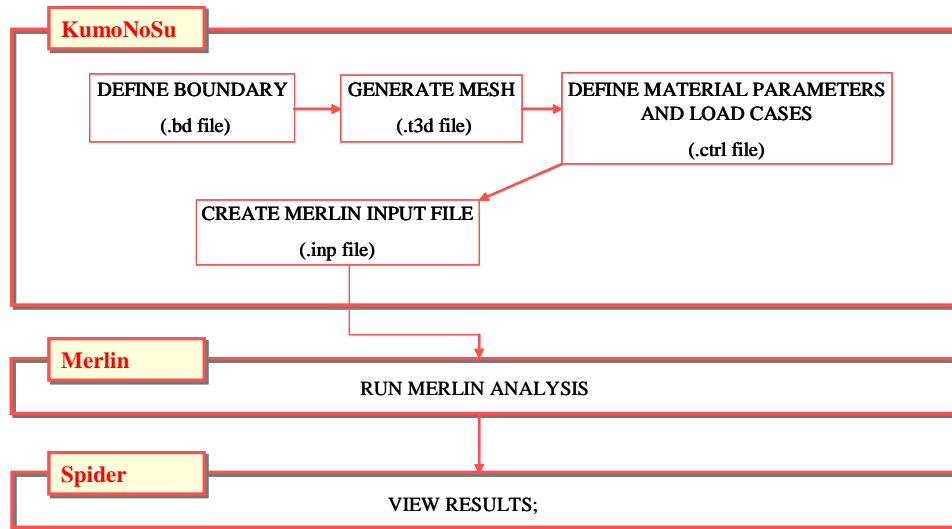


Figure 7.3: Program Interactions

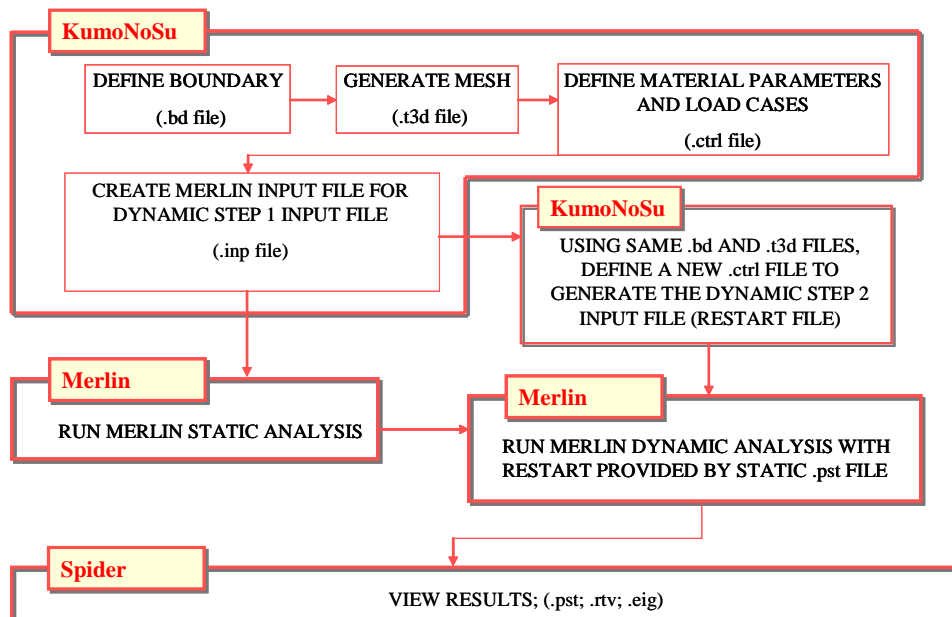


Figure 7.4: Program Interactions

Entity	Geometry	Topology
Region	free	-List of boundary surfaces, patches, and shells -List of fixed vertices, curves, surfaces, patches, and shells
Surface	Rational Bezier surface	4 bounding curves -List of fixed vertices, curves, and surfaces -Parent region or surface -Regions on both sides
Patch	Plane	-List of boundary curves -List of fixed vertices and curves -Parent region -Regions on both sides
Shell	Rational Bezier surface	-List of boundary curves -List of fixed vertices and curves -Parent region -Regions on both sides
Curve	Rational Bezier curve	-2 bounding vertices -List of fixed vertices and curves -Parent region, surface, patch, shell, or curve -List of connected surfaces, patches, and shells
Vertex	Point	-Parent region, surface, patch, shell, or curve -Parent vertex -List of fixed vertices -List of connected curves

Table 7.2: Hierarchy of Model Representation

Appendix A

MESH GENERATION

A.1 Introduction

Finite element mesh generation is now an integral part of a finite element analysis. With the increased computational capabilities, increasingly more complex structures are being analysed. Those structures must be discretized.

The task is one of developing a mathematical model (discretization or tessellation) of a continuum model. This is not only necessary in finite element analysis, but in computer graphics/rendering also. In computer graphics, we focus on the boundary representation, and assign colors and shades on the basis of light source and outward normal direction of the polygon.

Hence, in the most general case, meshing can be defined as the process of breaking up a physical domain into smaller sub-domains (elements) in order to facilitate the numerical solution of a partial differential equation. Surface domains may be subdivided into triangle or quadrilateral shapes, while volumes may be subdivided primarily into tetrahedra or hexahedra shapes. The shape and distribution of the elements is ideally defined by automatic meshing algorithms.

1. Point placement, followed by triangularization (discussed below).
2. Sub-domain removal. Elements are gradually removed from the domain, one at a time, until the whole domain is decomposed into finite elements.
3. Recursive subdivision. The domain is broken into simpler parts until the individual parts are single elements or simple regions, that can be meshed directly, for instance by the conformal mapping algorithm.
4. Hierarchical decomposition. The basic principle of a quadtree (or hierarchical decomposition) is to cover a planar region of interest by a square, then recursively partition squares into smaller squares until each square contains a suitably uniform subset of the input.

A.2 Triangulation

The concept of Voronoi diagrams first appeared in works of Descartes as early as 1644. Descartes used Voronoi-like diagrams to show the disposition of matter in the solar system and its environs.

The first man who studied the Voronoi diagram as a concept was a German mathematician G. L. Dirichlet. He studied the two- and three dimensional case and that is why this concept is also known as Dirichlet tessellation. However it is much better known as a Voronoi diagram because another German mathematician M. G. Voronoi in 1908 studied the concept and defined it for a more general n-dimensional case.

Very soon after it was defined by Voronoi it was developed independently in other areas like meteorology and crystallography. Thiessen developed it in meteorology in 1911 as an aid to computing more accurate estimates of regional rainfall averages. In the field of crystallography German researchers

dominated and Niggli in 1927 introduced the term *Wirkungsbereich* (area of influence) as a reference to a Voronoi diagram.

During the years this concept kept being rediscovered over and over again in different fields of science and today it is extensively used in about 15 different fields of sciences. Some of them being mathematics, computer science, biology, cartography, physiology and many others.

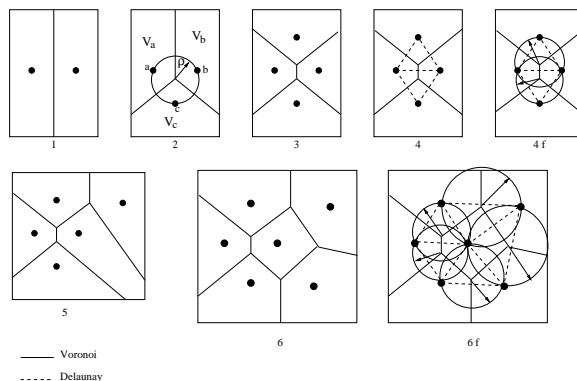


Figure A.1: Voronoi and Delaunay Tessellation

A.2.1 Voronoi Polygon

Given a finite set of points in the plane, the idea is to assign to each point a region of influence in such a way that the regions decompose the plane, (?). To describe a specific way to do that, let S be a subset of \mathbb{R}^2 ($S \subseteq \mathbb{R}^2$). We define the **Voronoi region** of $p \in S$ as the set of points $x \in \mathbb{R}^2$ that are at least as close to p as to any other points in S :

$$V_p = \{x \in \mathbb{R}^2 \mid \|x - p\| \leq \|x - q\|, \forall q \in S\} \quad (\text{A.1})$$

Each point $x \in \mathbb{R}^2$ has at least one nearest point in S , so it lies in at least one Voronoi region. Two Voronoi regions lie on opposite sides of the perpendicular bisector separating the two generating points.

A.2.2 Delaunay Triangulation

The dual of the Voronoi diagram is obtained by drawing straight *Delaunay edges* connecting points $p, q \in S$ if and only if their Voronoi regions intersect along a common line segment. Thus in general, the Delaunay edges decompose the convex hull of S into triangular regions which are referred to as **Delaunay triangles**, (?).

Using Euler's relation, it can be shown that a planar graph with $n \geq 3$ vertices has at most $3n - 6$ edges and at most $2n - 4$ faces. The same bounds hold for the number of Delaunay edges and triangles. Each Voronoi vertex $u = V_a \cap V_b \cap V_c$ is the center of a circle with radius $\rho = \|u - a\| = \|u - b\| = \|u - c\|$. The circle is **empty** because it encloses no point of S .

Additional detailed information on Voronoi tessellation, (?) is an excellent reference.

A.2.3 MATLAB Code

```
rand('state',4);
x = rand(1,3);
y = rand(1,3);
TRI = delaunay(x,y);
subplot(1,2,1),...
trimesh(TRI,x,y,zeros(size(x))); view(2),...
```

```
axis([0 1 0 1]); hold on;
plot(x,y,'o');
set(gca,'box','on');

[vx, vy] = voronoi(x,y,TRI);
subplot(1,2,2),...
plot(x,y,'r+',vx,vy,'b-'),...
axis([0 1 0 1])
```

A.3 Finite Element Mesh Generation

A.3.1 Boundary Definition

In order to discretize the continuum into a finite element mesh, first key geometrical information of the must be specified hierarchically:

Vertices: with nodal coordinates, and approximate desired element size in the immediate vicinity (thus describing the mesh density).

Edges: which connect vertices. Those can be either linear segments, polylines, or curves.

Surfaces: composed of edges, defined counterclockwise.

Volumes: (3D only) composed of surfaces.

Associated with surfaces (2D0, or volumes (3D) are different material properties.

Examples of the hierarchical boundary definition is shown in Fig. A.2 and A.3.

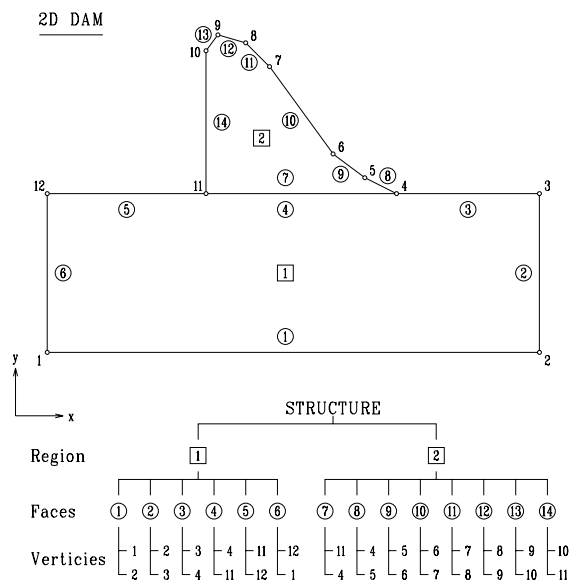


Figure A.2: Control Point for a 2D Mesh

A.3.2 Interior Node Generation

Once the boundary has been defined, we need to insert internal nodes at a spacing which respect the required mesh density. There are numerous techniques to insert those internal nodes. We present one approach, Fig. A.4

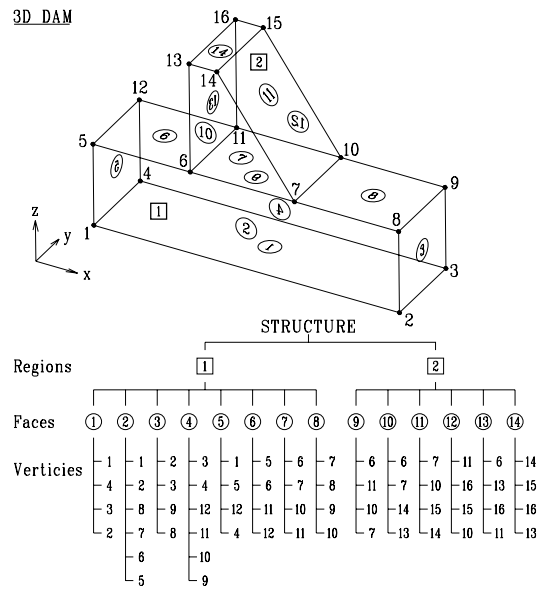


Figure A.3: Control Point for a 3D Mesh

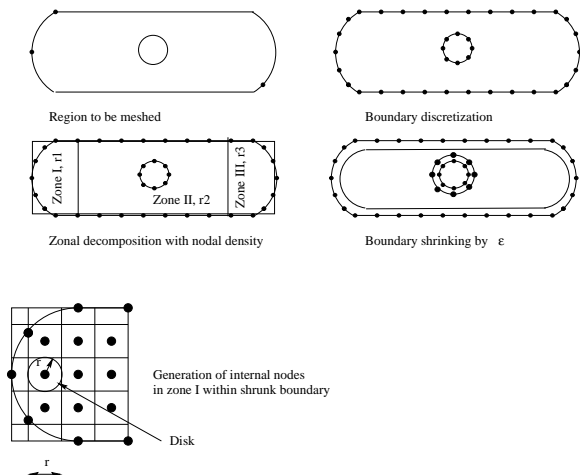


Figure A.4: A Two Dimensional Triangularization AlgorithmControl Point for a 3D Mesh

1. Decompose the region into a disjoint ensemble of subregions with equal mesh density.
2. Shrink the mesh to avoid elements near the boundary with very acute interior angles.
3. Starting with the first zone, circumscribe it by the smallest possible rectangle.
4. Superimpose a square rectangular grid over the circumscribing rectangle.
5. Use a random number generator to randomly generate one interior node in each square. A disk of radius r centered at each node is used to test that no other surrounding nodes are enclosed in the disk. If so, the node in question is regenerated.

An alternative approach consists in, (Červenka, J. 1994)

1. Generating a triangularization compatible with the initial nodes.
2. Check lengths of the edges. If an edge does not satisfy the prescribed size r , a new node is inserted in the center of the edge. The prescribed size is interpolated between those of the vertices at each end of the edge.
3. Repeat this operation until convergence.
4. Smoothen the elements to assure appropriate aspect ratios.

A.3.3 Final Triangularization

With boundary and interior nodes generated:

1. Determine the Voronoi polygons
2. Perform a Delaunay triangularization
3. Smoothen the mesh to ensure that all generated elements have a satisfactory aspect ratio.

It should be noted that recent algorithms, which can generate quadrilateral elements out of the Delaunay triangularization have recently emerged.

Appendix B

Rational Bezier Curve

The rational Bezier curve is determined by a control polygon, as shown in Fig. B.1.

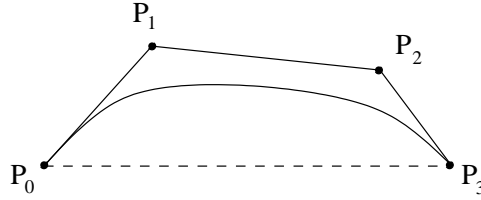


Figure B.1: Example of Rational Bezier Curve and Its Control Polygon

The curve generally follows the shape of the control polygon, the first and the last points on the curve are coincident with the first P_0 and the last point P_3 of the control polygon, and the the first and the last segments of the control polygon coincide with the curve tangent at the starting and the ending points of the curve, respectively.

Mathematically, it can be represented by the following equation(?).

$$P(t) = \frac{\sum_{i=0}^n w_i P_i B_i^n(t)}{\sum_{i=0}^n w_i B_i^n(t)}, \quad (\text{B.1})$$

where $P(t)$ is the point on the curve, P_i are Bezier control points, w_i are weight of Bezier control points, B_i^n is Bernstein polynomials, t is an independent variable varying in range from 0 to 1, and n is the degree of the polynomial defining the curve segment.

Bernstein polynomial function is defined as

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad (\text{B.2})$$

or recursively as

$$B_i^n(t) = (1-t) B_i^{n-1}(t) + t B_{i-1}^{n-1}(t), \quad (\text{B.3})$$

where $B_0^0 = 1$.

The curve order is equal to the number of polygon points $n + 1$. Therefore, a 4 point rational bezier polygons results in a cubic curve whereas a 3 point one results in a quadratic curve.

In the mesh generator program, the point P_0 and P_n correspond to the model vertices while the remaining points form the control polygon of the curve. In the particle model, the elliptical curve has been used to define the inclusion. Therefore, to construct the rational Bezier curve the control points of the elliptical curve must be determined based on the degree of the polynomial of the elliptical curve.

B.1 Quadratic curve

In the $T3D$ (?) mesh generator used internally by *Kumo*(?), the quadratic curve of the elliptical arc is defined as demonstrated in Fig. B.2.

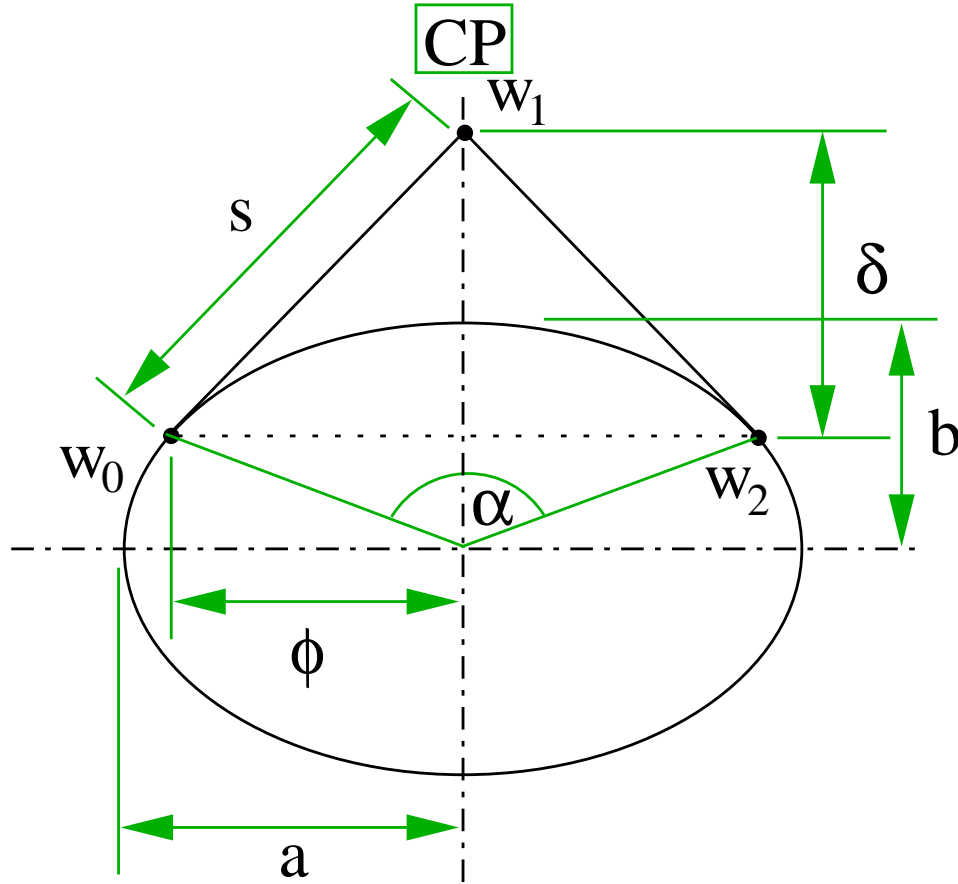


Figure B.2: Elliptical Arc-Quadratic Curve

Equations for each the variables shown in Fig. B.2 can be summarized as follows.

$$\delta = b \frac{\sin^2 \frac{\alpha}{2}}{\cos \frac{\alpha}{2}} \quad (\text{B.4})$$

$$\phi = a \sin \frac{\alpha}{2} \quad (\text{B.5})$$

$$s = \frac{\sin \frac{\alpha}{2}}{\cos \frac{\alpha}{2}} \sqrt{a^2 \cos^2 \frac{\alpha}{2} + b^2 \sin^2 \frac{\alpha}{2}} \quad (\text{B.6})$$

$$w_0 = w_2 = 1 \quad (\text{B.7})$$

$$w_1 = \cos \frac{\alpha}{2} \quad (\text{B.8})$$

$$\alpha \in (0 : 2\pi) \setminus \{\pi\} \quad (\text{B.9})$$

where w_0 , w_1 , and w_2 are weights of the bezier control points.

The y coordinate of the control point CP is then determined by

$$y_{CP} = \frac{\phi}{\tan \frac{\alpha}{2}} + \delta \quad (\text{B.10})$$

In the general case where the starting and the ending points are not in the symmetrical pattern shown in Fig. B.2, Fig.B.3, determination of the control point and its weight requires special attention.

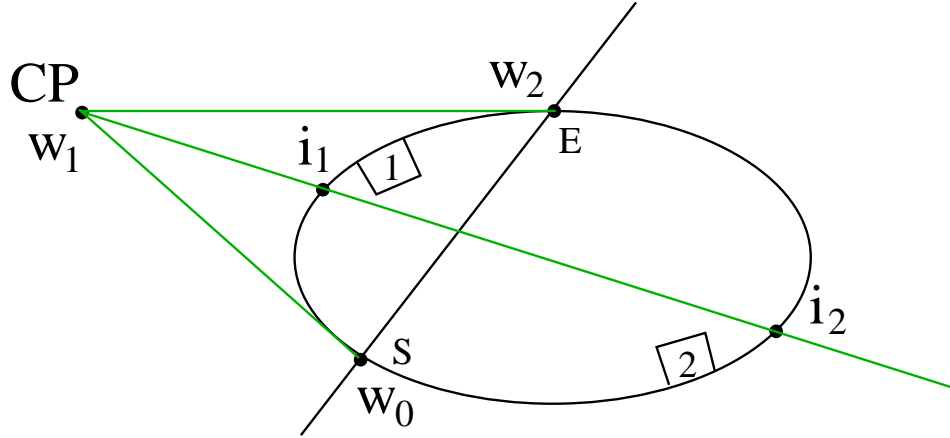


Figure B.3: General Case of Elliptical Arc-Quadratic Curve

Assuming that S is the starting point of the curve 1 (or 2) and E is the ending point of the curve 1 (or 2). To determine the control point of both curve 1 and 2, first the tangent to the elliptical curve at the starting S and the ending E point are drawn and the intersection of the 2 tangent lines is the control point CP . It was shown by (?) that the weight w_0 and w_2 of both curve 1 and 2 are the same and are equal to 1. The weight w_1 for both curve can be determined by following equations.

1. Curve 1

$$w_1 = \frac{M - I_1}{I_1 - CP} \quad (\text{B.11})$$

2. Curve 2

$$w_2 = \frac{M - I_2}{I_2 - CP} \quad (\text{B.12})$$

where M is the coordinate of the mid point of the line SE , I_1 is the coordinate of the intersection point between the elliptical curve 1 and the line originating from the control point passing through point M and I_2 is the coordinate of the intersection point between the elliptical curve 2 and the line originating from the control point passing through point M .

B.2 Cubic curve

In the $T3D$ mesh generator (?), the cubic curve of the elliptical arc is defined in Fig. B.4.

All equations of each variables shown in Fig. B.4 are summarized as follows.

$$\delta = 2b \frac{\sin^2 \frac{\alpha}{2}}{1 + 2 \cos \frac{\alpha}{2}} \quad (\text{B.13})$$

$$\phi = a \frac{\sin \alpha}{1 + 2 \cos \frac{\alpha}{2}} \quad (\text{B.14})$$

$$s = \frac{2 \sin \frac{\alpha}{2}}{1 + 2 \cos \frac{\alpha}{2}} \sqrt{a^2 \cos^2 \frac{\alpha}{2} + b^2 \sin^2 \frac{\alpha}{2}} \quad (\text{B.15})$$

$$w_0 = w_3 = 1 \quad (\text{B.16})$$

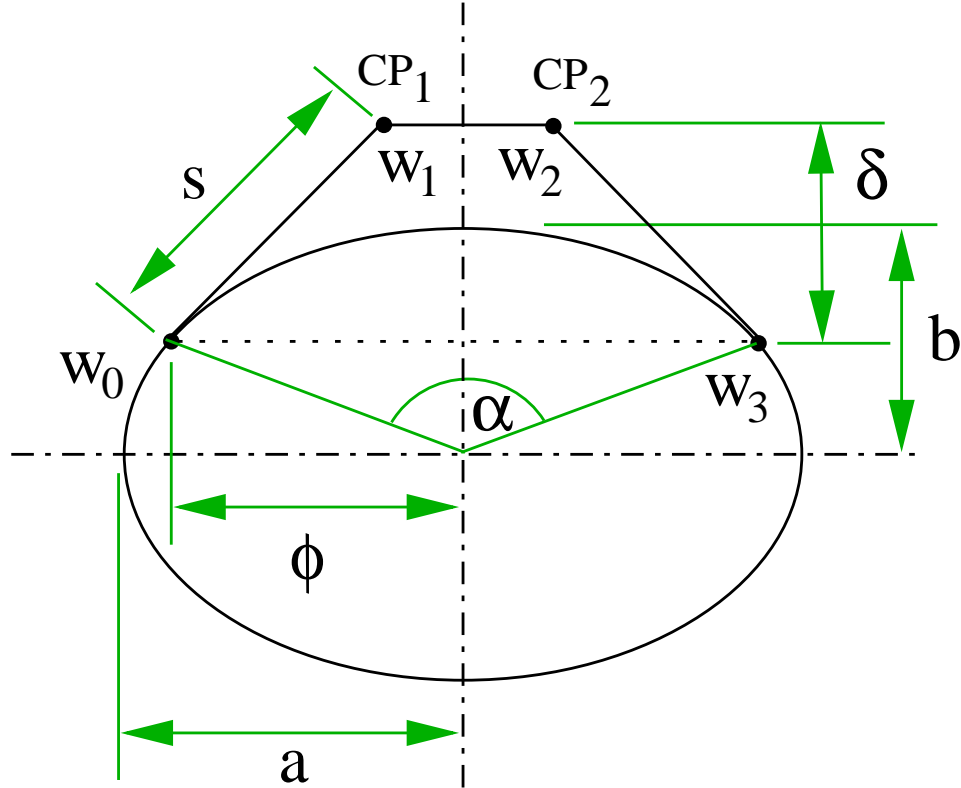


Figure B.4: Elliptical Arc-Cubic Curve

$$w_1 = w_2 = \frac{1 + 2 \cos \frac{\alpha}{2}}{3} \quad (\text{B.17})$$

$$\alpha \in (0 : 2\pi) \setminus \left\{ \frac{4}{3}\pi \right\} \quad (\text{B.18})$$

where w_0, w_1, w_2 and w_3 are weights of the Bezier control points.

The control points CP_1 and CP_2 are the points which have the distance s from starting point SP and the ending point EP in the direction of the tangent line to the curve at the starting and the ending points, respectively.

In the case where the starting and the ending points are not in the symmetrical pattern as shown in Fig. B.5, and both points are opposite to each other as shown in Fig.B.5.

The value of the weights of the Bezier curve are still the same as shown in Eq. B.16 and B.17. However, the coordinates of both control points CP_1 and CP_2 can be computed as follows.

From Fig.B.5, suppose SP and EP are the starting and the ending points of curve 1.

1. Draw the tangent line to the elliptical curve at the SP and EP and determine the slope of the tangent s_1 .
2. Draw a line from the origin O , which a slope equal to s_1 . This line intersects with curve 1 at point P .
3. Draw a tangent line to the elliptical curve 1 at point P (slope = s_2) until it intersects the tangent line originating from the starting and the ending points at point A and B , respectively.
4. Determine the distance d .

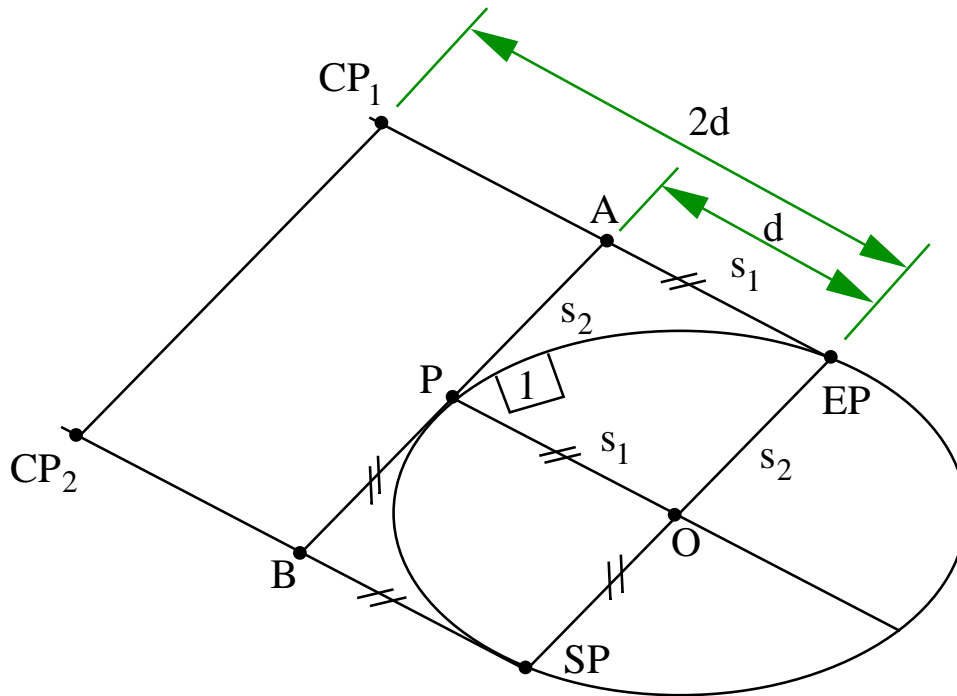


Figure B.5: Specific Case of Elliptical Arc-Cubic Curve

- Control points CP_1 and CP_2 are then the points from the starting and the ending points with distance $2d$ in the direction of its tangent, respectively.

Appendix C

Examples of Finite Element Boundary Definition Output Files from PARSIFAL Program

C.1 Matrix and Inclusion without Interface elements

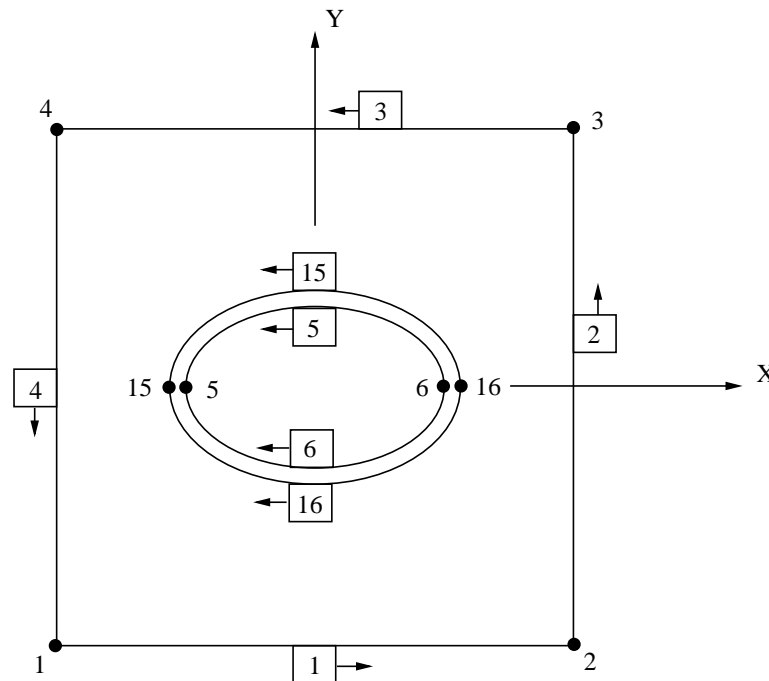


Figure C.1: Matrix and inclusion without interface elements

Fig.C.1 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with a perfectly rigid interface (outside ellipse) between the matrix and inclusion. The boundary description for this problem is defined using (8) vertices, (8) curves, (3) patches, and (4) master/slave relations. The resulting .bd file can be shown as follows.

```

vertex 1 xyz -1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 2 xyz 1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 3 xyz 1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 4 xyz -1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 5 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 6 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 15 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 16 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000

curve 1 order 2 vertex 1 2
curve 2 order 2 vertex 2 3
curve 3 order 2 vertex 3 4
curve 4 order 2 vertex 4 1
curve 5 order 4 vertex 6 5
polygon 1 xyz 5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 6 order 4 vertex 6 5
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 15 order 4 vertex 16 15
polygon 1 xyz 5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 16 order 4 vertex 16 15
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002 property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 15 -16
size def hole
patch 3 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

masterslave 1 vertex 5 15
masterslave 2 vertex 6 16
masterslave 3 curve 5 15
masterslave 4 curve 6 16

```

Note that vertices 5 and 15 share the same coordinates, as do vertices 6 and 16. Additionally, curves 5/15 and 6/16 share identical properties. However, these curves combine to form separate patches, with curves 5,6 forming patch 1 and curves 15,16 forming patch 2. Additionally, patch 2 is defined as a hole, since it is simply providing the connection between the inclusion (patch 1) and the matrix (patch 3). Patch 2 serves as a subpatch for patch 3.

In order to enforce the rigid connection between the matrix and the inclusion, master/slave relationships are defined for vertices 5/15 and 6/16, and for curves 5/15 and 6/16. These relations effectively

join the matrix and inclusion in the resulting MERLIN .inp file.

C.2 Matrix and Inclusion with Interface Elements

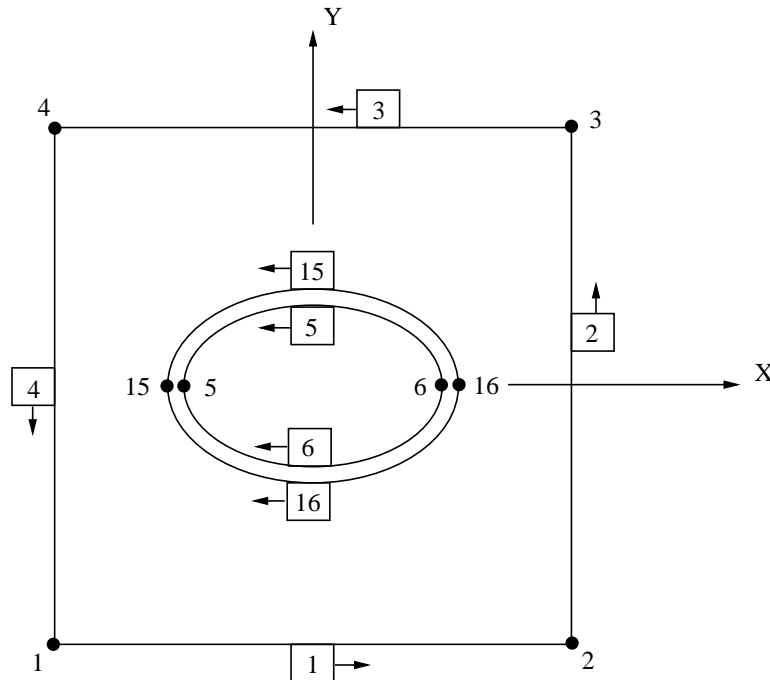


Figure C.2: Matrix and inclusion with interface elements

Fig.C.2 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with interface elements (outside ellipse) between the matrix and inclusion. The boundary description for this problem is defined using (8) vertices, (8) curves, (3) patches, (2) crack segments, and (2) crack paths. The resulting .bd file can be shown as follows.

```
# This input file generated by T3d preprocessor at 15:56:28 on 11/05/01
```

```
vertex 1 xyz -1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 2 xyz 1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 3 xyz 1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 4 xyz -1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 5 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 6 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 15 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 16 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000
```

```
curve 1 order 2 vertex 1 2
curve 2 order 2 vertex 2 3
curve 3 order 2 vertex 3 4
curve 4 order 2 vertex 4 1
curve 5 order 4 vertex 6 5
polygon 1 xyz 5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
```

```

weight 3.333e-001
curve 6 order 4 vertex 6 5
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 15 order 4 vertex 16 15
polygon 1 xyz 5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 16 order 4 vertex 16 15
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002 property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 15 -16
size def hole
patch 3 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

crack NLFM
crack 1 structure curve 15 5
crack 2 structure curve 16 6
crack structure path 1 property 3
crack structure path 2 property 3

```

Note that vertices 5 and 15 share the same coordinates, as do vertices 6 and 16. Additionally, curves 5/15 and 6/16 share identical properties. However, these curves combine to form separate patches, with curves 5,6 forming patch 1 and curves 15,16 forming patch 2. Additionally, patch 2 is defined as a hole, since it is simply providing the connection between the inclusion (patch 1) and the matrix (patch 3). Patch 2 serves as a subpatch for patch 3.

In order to insert the interface elements between the matrix and the inclusion, structure crack segments are defined between curves 5/15 and 6/16, and a two crack paths are defined using these two crack segments. Since the crack type is defined as NLFM, interface elements will be inserted between the matrix and inclusion in the MERLIN .inp file.

C.3 Matrix, Inclusion, and propagating Crack with Interface Elements

Fig.C.3 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with interface elements (outside ellipse) between the matrix and inclusion. Additionally, a crack extends from the right side of the inclusion (triangle). This crack, while represented by a triangle with a finite crack mouth opening in the above picture, is considered to have a zero thickness in the .bd file. The boundary description for this problem is defined using (10) vertices, (10) curves, (3) patches, (3) crack segments, and (2) crack paths. The resulting .bd file can be shown as follows.

```

# This input file generated by T3d preprocessor at 12:47:58 on 11/06/01

vertex 1 xyz -1.00000000e+000 -1.00000000e+000 0.00000000e+000

```

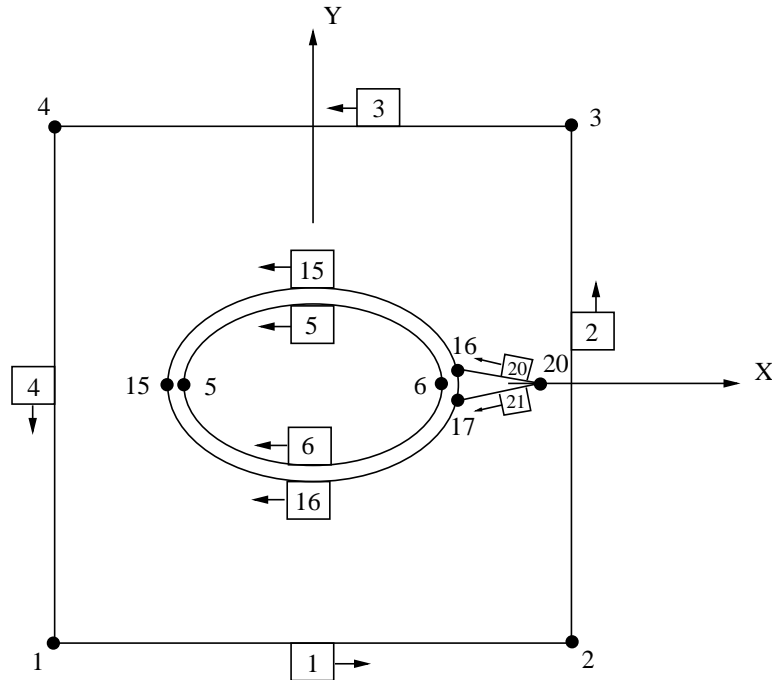


Figure C.3: Matrix, inclusion, and propagating crack with interface elements

```

vertex 2 xyz 1.0000000e+000 -1.0000000e+000 0.0000000e+000
vertex 3 xyz 1.0000000e+000 1.0000000e+000 0.0000000e+000
vertex 4 xyz -1.0000000e+000 1.0000000e+000 0.0000000e+000
vertex 5 xyz -5.0000000e-001 0.0000000e+000 0.0000000e+000
vertex 6 xyz 5.0000000e-001 0.0000000e+000 0.0000000e+000
vertex 15 xyz -5.0000000e-001 0.0000000e+000 0.0000000e+000
vertex 16 xyz 5.0000000e-001 0.0000000e+000 0.0000000e+000
vertex 17 xyz 5.0000000e-001 0.0000000e+000 0.0000000e+000
coincide vertex 16
vertex 20 xyz 7.5000000e-001 0.0000000e+000 0.0000000e+000

curve 1 order 2 vertex 1 2
curve 2 order 2 vertex 2 3
curve 3 order 2 vertex 3 4
curve 4 order 2 vertex 4 1
curve 5 order 4 vertex 6 5
polygon 1 xyz 5.0000000e-001 2.5000000e-001 0.0000000e+000
weight 3.333e-001
polygon 2 xyz -5.0000000e-001 2.5000000e-001 0.0000000e+000
weight 3.333e-001
curve 6 order 4 vertex 6 5
polygon 1 xyz 5.0000000e-001 -2.5000000e-001 0.0000000e+000
weight 3.333e-001
polygon 2 xyz -5.0000000e-001 -2.5000000e-001 0.0000000e+000
weight 3.333e-001
curve 15 order 4 vertex 16 15
polygon 1 xyz 5.0000000e-001 2.5000000e-001 0.0000000e+000
weight 3.333e-001

```

```

polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 16 order 4 vertex 17 15
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 20 vertex 20 16
curve 21 vertex 20 17 coincide curve 20

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002 property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary
curve 20 15 -16 -21 size def hole
patch 4 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

crack NLFM
crack 1 structure curve 21 20 front2d 20
crack 2 structure curve 5 15
crack 3 structure curve 16 6
crack structure path 1 mat 3
crack structure path 2 3 mat 3

```

Note that vertices 5 and 15 share the same coordinates, as do vertices 6, 16, and 17. Vertices 16 and 17, which form the mouth of the crack, are labeled as coincident. The crack propagating from the inclusion is formed using curves 20 and 21 (coincident), with vertex 20 serving as the crack front. Curves 5/15 and 6/16 also share identical properties. However, these curves combine to form separate patches, with curves 5,6 forming patch 1 and curves 15,16 (with curves 20 and 21) forming patch 2. Additionally, patch 2 is defined as a hole, since it is providing the interface connection between the inclusion (patch 1) and the matrix (patch 3). Patch 2 serves as a subpatch for patch 3.

Two separate crack paths are considered. The first is the crack propagating from the inclusion. This crack path is defined by a single crack segment formed by curves 20 and 21, with the crack front at vertex 20. The second crack path surrounds the inclusion, and is formed by two crack segments (curves 5/15 and 6/16).

C.4 Matrix, Inclusion, and Two Propagating Cracks with Interface Elements

Fig.C.4 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with interface elements (outside ellipse) between the matrix and inclusion. Additionally, cracks extends from the right and left sides of the inclusion (triangle). This crack, while represented by a triangle with a finite crack mouth opening in the above picture, is considered to have a zero thickness in the .bd file. The boundary description for this problem is defined using (12) vertices, (12) curves, (3) patches, (4) crack segments, and (4) crack paths. The resulting .bd file can be shown as follows.

```

# This input file generated by Kumonosu at 11:48:44 on 11/07/01

vertex 1 xyz -1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 2 xyz 1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 3 xyz 1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 4 xyz -1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 5 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 6 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000

```

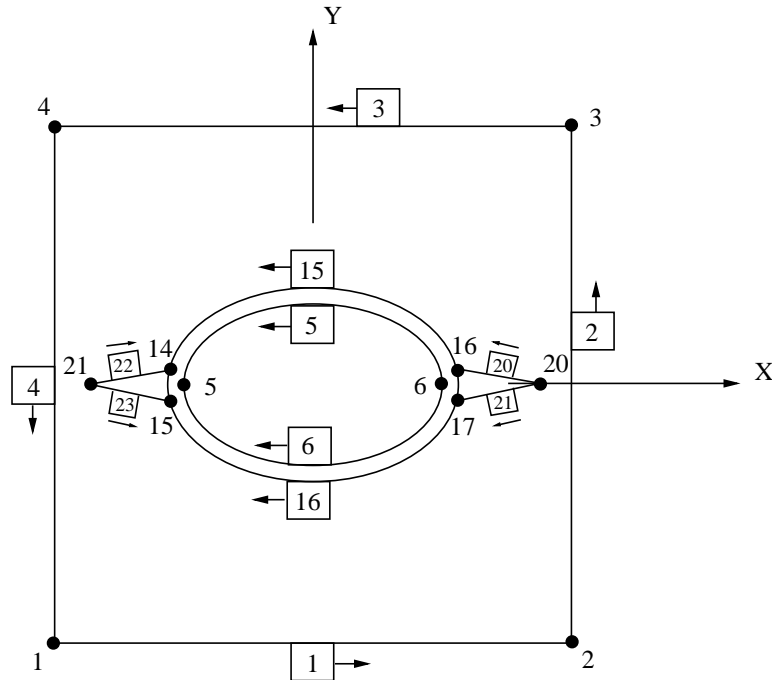


Figure C.4: Matrix, inclusion, and two propagating cracks with interface elements

```

vertex 14 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 15 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
coincide vertex 14
vertex 16 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 17 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000
coincide vertex 16
vertex 20 xyz 7.50000000e-001 0.00000000e+000 0.00000000e+000
vertex 21 xyz -7.50000000e-001 0.00000000e+000 0.00000000e+000

curve 1 order 2 vertex 1 2
curve 2 order 2 vertex 2 3
curve 3 order 2 vertex 3 4
curve 4 order 2 vertex 4 1
curve 5 order 4 vertex 6 5
polygon 1 xyz 5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 6 order 4 vertex 6 5
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 15 order 4 vertex 16 14
polygon 1 xyz 5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 2.50000000e-001 0.00000000e+000
weight 3.333e-001

```

```

curve 16 order 4 vertex 17 15
polygon 1 xyz 5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001 -2.50000000e-001 0.00000000e+000
weight 3.333e-001
curve 20 vertex 20 16
curve 21 vertex 20 17 coincide curve 20
curve 22 vertex 21 14
curve 23 vertex 21 15 coincide curve 22

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002 property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve
20 15 -22 23 -16 -21 size def hole
patch 4 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

crack NLFM
crack 1 structure curve 21 20 front2d 20
crack 2 structure curve 22 23 front2d 21
crack 3 structure curve 5 15
crack 4 structure curve 16 6
crack structure path 1 property 3
crack structure path 2 property 3
crack structure path 3 property 3
crack structure path 4 property 3

```

Unlike the example with a single propagating crack, in which the interface around the inclusion was treated as a single crack path, this geometry requires that (4) crack paths are defined: one for the left propagating crack, one for the right propagating crack, and two paths around the inclusion (top and bottom). The rest of the boundary description is similar to that of the single crack example. There are (3) total patches: one for the inclusion, a second composed of the inclusion and the two cracks (defined as a hole), and a third which is composed of the matrix.

C.5 Matrix and Interior Crack with Interface Elements

Fig. C.5 illustrates an interior crack (red lines) surrounded by a matrix (yellow box). The boundary description for this problem is defined using (8) vertices, (8) curves, (1) patch, (2) crack segments, and (2) crack paths. The resulting .bd file is on the next page.

```

# This input file generated by Kumonosu at 14:21:05 on 11/29/01

vertex 1 xyz -1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 2 xyz 1.00000000e+000 -1.00000000e+000 0.00000000e+000
vertex 3 xyz 1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 4 xyz -1.00000000e+000 1.00000000e+000 0.00000000e+000
vertex 5 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 6 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000
vertex 7 xyz 0.00000000e+000 0.00000000e+000 0.00000000e+000
vertex 8 xyz 0.00000000e+000 0.00000000e+000 0.00000000e+000 coincide vertex 7

curve 1 order 2 vertex 1 2
curve 2 order 2 vertex 2 3

```

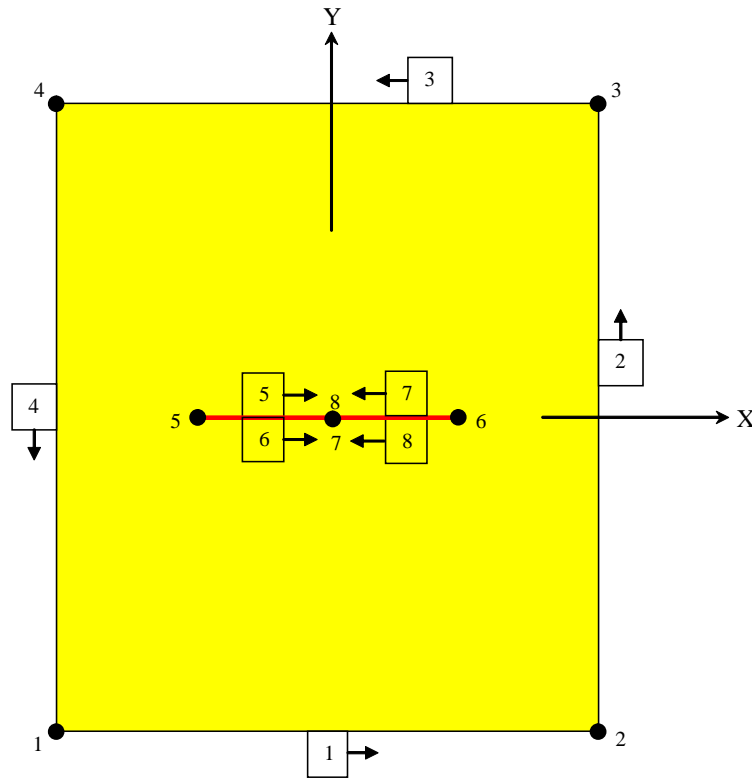


Figure C.5: Matrix, Crack inclusion with interface elements

```

curve 3 order 2 vertex 3 4
curve 4 order 2 vertex 4 1
curve 5 order 2 vertex 5 8
curve 6 order 2 vertex 5 7 coincide curve 5
curve 7 order 2 vertex 6 8
curve 8 order 2 vertex 6 7 coincide curve 7

patch 1 normal 0.00000000e+000 0.00000000e+000 1.00000000e+000 boundary curve 1 2 3 4 \
      boundary curve -5 6 -8 7 size 1.000e-001 property 1

crack NLFM
crack 1 structure curve 5 6 front2d 5
crack 2 structure curve 8 7 front2d 6
crack structure path 1 property 2
crack structure path 2 property 2

vertex 5000 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000 coincide vertex 5 slave node
vertex 5001 xyz 0.00000000e+000 0.00000000e+000 0.00000000e+000 coincide vertex 8 slave node
vertex 5002 xyz -5.00000000e-001 0.00000000e+000 0.00000000e+000 coincide vertex 5 slave node
vertex 5003 xyz 0.00000000e+000 0.00000000e+000 0.00000000e+000 coincide vertex 7 slave node
vertex 5004 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000 coincide vertex 6 slave node
vertex 5005 xyz 0.00000000e+000 0.00000000e+000 0.00000000e+000 coincide vertex 7 slave node
vertex 5006 xyz 5.00000000e-001 0.00000000e+000 0.00000000e+000 coincide vertex 6 slave node
vertex 5007 xyz 0.00000000e+000 0.00000000e+000 0.00000000e+000 coincide vertex 8 slave node

curve 5000 vertex 5000 5001 order 2 coincide curve 5 slave element
curve 5001 vertex 5002 5003 order 2 coincide curve 6 slave element
curve 5002 vertex 5004 5005 order 2 coincide curve 8 slave element
curve 5003 vertex 5006 5007 order 2 coincide curve 7 slave element

```

Notice that the single interior crack is actually composed of two cracks. Curves 5/6 form one side of the crack, while curves 7/8 form the other side. Vertices 5 and 6 serve as the crack fronts, and vertices 7 and 8 (coinciding vertices) form the middle of the crack. An interior crack must have these coinciding vertices at its center. This boundary description contains only one patch, but this patch contains two definitions of boundary curves. The first set of boundary curves defines the edges of the matrix, while the second set of boundary curves defines the crack.